UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA TELEKOMUNIKACIJE

Razvoj elektroničke strukture za daljinsko upravljanje kapijom

ZAVRŠNI RAD

- PRVI CIKLUS STUDIJA -

Student:
Harun Muhić

Mentor:
Doc. dr Emir Sokić, dipl.ing.el.

Sarajevo,
septembar 2018.

Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za telekomunikacije
Doc. dr Emir Sokić, dipl.el.ing.
Sarajevo, 24.09.2018.

Tema za završni rad

studenata I ciklusa studija koji studiraju na ETF-u u skladu sa principima Bolonjskog procesa na Odsjeku za telekomunikacije (ak.g. <2017/18>)

Tema: Razvoj elektroničke strukture za daljinsko upravljanje kapijom
Student: Harun Muhić

Tema:
Mogućnost daljinskog upravljanja ulaznom kapijom ili garažnim vratima na privatnim i poslovnim objektima su već duži vremenski period komercijalno dostupno u praksi. Većina ovakvih sistema sadrži nekoliko osnovnih dijelova: modul za daljinsko upravljanje (tipično baziran na RF 315/433MHz), aktuatore (linearne ili rotacione) za pomjeranje kapije/vrata, svjetlosne i zvučne indikatore, senzore (za onemogućavanje neželjenog zatvaranja/otvaranja u slučaju kvara ili prisustva ljudi, te detekciju da je kapija zatvorena/otvorena) i upravljački modul koji je zadužen za mjerenje, komunikaciju i upravljanje cjelokupnim sistemom. Uzimajući u obzir nagli razvoj IoT, u okviru ovog završnog rada potrebno je razviti elektroničku strukturu koja omogućava da se upravljanje i nadzor kapije obavlja korištenjem Wi-Fi pristupa i odgovarajuće mobilne aplikacije.

Postavka zadatka:
U okviru rada potrebno je:

- dati osnovni pregled literature,
- osmisliti i dizajnirati elektronički sklop koji omogućava upravljanje i nadzor kapije putem WiFi pristupa,
- testirati prototip u simulacionom okruženju,
- dizajnirati i izraditi odgovarajuću elektroničku ploču,
- napraviti minijaturnu maketu kapije koja omogućava testiranje sistema,
- (opcionalno) razviti jednostavnu mobilnu aplikaciju koja omogućava upravljanje sistemom,
- testirati razvijeni sistem.

Koncept i metode rješavanja:
Rad se treba sastojati iz sljedećih cjelina:

- Pregled literature i analiza postojećih komercijalnih rješenja,
- Analiza mogućih pristupa kod WiFi-baziranog upravljanja i razvoj idejnog rješenja,
- Dizajniranje elektroničke strukture korištenjem odgovarajućih softverskih alata (PCB Eagle, Proteus i sl.),
- Izrada odgovarajuće elektroničke strukture,
- (opcionalno) Izrada mobilne aplikacije,
- Eksperimentalna analiza.

Osnovna literatura:

[1] Fraden, Jacob. Handbook of modern sensors: physics, designs, and applications. Springer Science & Business Media, 2004.

[2] Morris, Alan S. "Measurement and instrumentation principles." (2001): 1743.

[3] McRoberts, Michael, Brad Levy, and Cliff Wootton. Beginning Arduino. New York.: Apress, 2010.

[4] Oxer, Jonathan, and Hugh Blemings. Practical Arduino: cool projects for open source hardware. Apress, 2011.

[5] Espressif IoT Team, ESP8266 Technical Reference, 2017

[6] Ed Burnette "Hello, Android: Introducing Google's Mobile Development Platform", Pragmatic Bookshelf, 2013.

Doc. dr Emir Sokić, dipl. ing. el.

Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za telekomunikacije

Izjava o autentičnosti radova

Završni rad
I ciklus studija

Ime i prezime: Harun Muhić
Naslov rada: Razvoj elektroničke strukture za daljinsko upravljanje kapijom
Vrsta rada: Završni rad I ciklusa studija
Broj stranica: 56

Potvrđujem:

- da sam pročitao dokumente koji se odnose na plagijarizam, kako je to definirano Statutom Univerziteta u Sarajevu, Etičkim kodeksom Univerziteta u Sarajevu i pravilima studiranja koja se odnose na I i II ciklus studija, integrirani studijski program I i II ciklusa i III ciklus studija na Univerzitetu u Sarajevu, kao i uputama o plagijarizmu navedenim na web stranici Univerziteta u Sarajevu;
- da sam svjestan univerzitetskih disciplinskih pravila koja se tiču plagijarizma;
- da je rad koji predajem potpuno moj, samostalni rad, osim u dijelovima gdje je to naznačeno;
- da rad nije predat, u cjelini ili djelimično, za stjecanje zvanja na Univerzitetu u Sarajevu ili nekoj drugoj visokoškolskoj ustanovi;
- da sam jasno naznačio prisustvo citiranog ili parafraziranog materijala i da sam se referirao na sve izvore;
- da sam dosljedno naveo korištene i citirane izvore ili bibliografiju po nekom od preporučenih stilova citiranja, sa navođenjem potpune reference koja obuhvata potpuni bibliografski opis korištenog i citiranog izvora;
- da sam odgovarajuće naznačio svaku pomoć koju sam dobio pored pomoći mentora i akademskih tutora/ica.

Sarajevo, 24. septembar 2018.

Harun Muhić

Sažetak

U radu je opisan postupak dizajna i razvoja elektroničke strukture za daljinsko upravljanje kapijom. Prvo je dat pregled postojećih komercijalnih rješenja. Potom je opisan način komunikacije sa elektroničkom strukturom za upravljanje kapijom pomoću daljinskog upravljača, tastature i mobilne aplikacije. Prezentirana je hardverska i softverska realizacija sistema, te su ukratko navedene osnovne teorijske postavke korištenih načina komunikacije. Za realizaciju elektroničke strukture korišten je ESP32 mikrokontroler i razvijena je Android aplikacija. Nakon realizacije svih funkcionalnosti napravljen je minijaturni model kapije kako bi se pokazala ispravnost elektroničke strukture.

Abstract

The aim of this work is to give insight into all aspects of the design and development of a electrial structure for gate remote controlling. Firstly, brief insight is given about commercial solutions. Then there is description about possible ways of communication with electrical structure for gate remote controlling using remote controler, keypad and mobile application. Hardware and software implementation of the system are presented, and there are descriptions about theoretical fundamentals of implemented communications channels. ESP32 microcontroller was used for implementation of electrical structure and Android application was developed. After realisation of all functionalities, miniature model of gate was constructed to show possibilities of electrical structure.

Sadržaj

Popis slika

Popis tabela

Poglavlje 1

Uvod

1.1 Obrazloženje teme

U današnjem svijetu postoje težnje ka tome da se što više uređaja iz naše okoline modifikuju u pametne uređaje. Može se primijetiti nagli razvoj IoT-a (eng. *Internet of Things*). Želja je da se svaki uređaj poveže na internet i da se prikupljaju podaci o radu uređaja. Na taj način se poboljšavaju njegove performanse. Jedan od razloga zašto se teži ka umrežavanju uređaja je lakše upravljanje i korištenje istih.

Elektronički sistemi sa mogućnošću upravljanja ulaznom kapijom ili garažnim vratima koriste se već duži niz godina. Za upravljanje ulaznom kapijom (garažnim vratima) obično se koriste linearni aktuatori za klizne kapije i rotacioni aktuatori za krilne kapije. Upravljanje se najčešće izvršava pomoću RF primopredajnika.

Elektronički sistemi za upravljanje kapijom sastoje se od sljedećih dijelova:

- aktuatora (motora),
- upravljačkog modula,
- senzorskih sistema,
- svjetlosnog i zvučnog indikatora,
- uređaja za komunikaciju.

Za kretnju kapije koriste se DC motori. Da bi se upravljalo DC motorima koriste se drajveri. Drajveri su elektroničke komponente koje imaju zadatak da pojačaju upravljačke signale. Oni malu ulaznu snagu pojačavaju na veću vrijednost kako bi se moglo upravljati motorom korištenjem mikrokontrolera. U ovom završnom radu nema potrebe za upravljanjem motorom sa više različitih brzina i zbog toga neće biti korišteni enkoderi.

Upravljački modul služi za prikupljanje podataka od senzorskih sistema, za upravljanje aktuatorom, svjetlosnim i zvučnim indikatorom i uređajima za komunikaciju. Mogući su razni načini implementacije upravljačkih modula. Često se koriste mikrokontroleri.

Senzorski sistemi se sastoje od više senzora koji prikupljaju podatke iz okoline. Koriste se optički senzori, ultrazvučni senzori, senzori pokreta sa ciljem da se izvrši detekcije ljudi, životinja, objekata. Postoji opasnost da će upravljanjem kapije sa udaljenosti dovesti do povrede ljudi, životinja ili uništavanja imovine i zbog toga se koriste senzori za detekciju prepreka prilikom kretnje kapije. Svjetlosni i zvučni indikatori se koriste za signalizaciju kako ne bi došlo do neželjenih posljedica.

U već postojećim rješenjima u svrhu komuniciranja sa upravljačkim modulom sa udaljenosti koriste se razni načini komunikacije. Koriste se Wi-Fi moduli, gdje se pomoću aplikacija upravlja sa kapijom. Drugi mogući način komunikacije je korištenje RF primopredajnika. Moguća je komunikacija i pomoću bluetooth uređaja, ali sa manjih udaljenosti u odnosu na dva prethodna načina komuniciranja.

1.2 Pregled stanja u oblasti istraživanja

Već duži period postoje rješenja za upravljanje kapijom koja su komercijalno dostupna. Kompanije koje proizvode modele za upravljanje kapijom su *RIB Srl, Mighty Mule i Elsema*. Svaki od navedenih proizvođača ima široku ponudu rješenja. Posmatrat će se modeli *K800 (RIB Srl), MM-SL2000B (Mighty Mule) i iS600 (Elsema)* zbog sličnosti sa planiranim rješenjem. Modeli će se porediti uzimajući u obzir način upravljanja, udaljenost sa koje se mogu upravljati, senzore i komponente za sigurnost i cijene modela.

Model	Senzori i komponente za sigurnost	*Upravljanje i komunikacija*	*Domet*	*Cijena*
K800	fotoćelija, mehanička sigurnosna traka, mehanički graničnici	RF upravljač, tastatura	do 150 m	800 KM
MM-S2000B	fotoćelija, graničnici	RF upravljač, tastatura	do 30 m	1500 KM
iS600	fotoćelija	RF upravljač	do 100 m	1100 KM

Tabela 1.1: Osnovne karakteristike postojećih modela za upravljanje kapijom

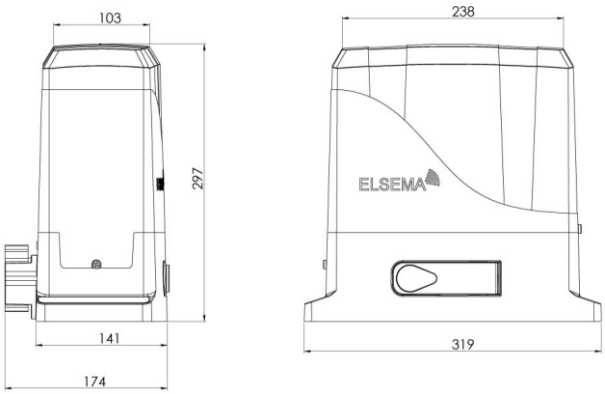
Za model *K800* postoji mogućnost upravljanja pomoću Wi-Fi modula. Moguće je i upravljanje određenim komponentama pomoću Wi-Fi master uređaja (domet 20 m). Ovaj način olakšava instalaciju uređaja zbog mogućnosti povezivanja velikog broja komponenti bežičnim putem [12].

Kod modela *MM-S2000B* moguće je korištenje univerzalnog prijemnika. Univerzalni prijemnik ima mogućnost da komunicira sa predajnikom bez obzira na marku proizvođača (domet do 30 m). Može se koristiti senzor za vozila koji automatski aktivira upravljački modul. U ponudi se još nalazi bežični alarm za prilaz, koji šalje informacije ukoliko se vozilo približi kapiji [13].

Za model *iS600* postoje razni dodaci za komunikaciju, kao što je GSM otvarač kapije kako bi se omogućilo upravljanje sa mobilnim telefonom ili *PentaCODE* tastatura za upravljanje bežičnim putem ukoliko se koristi odgovarajući prijemnik.

Za rad modela *iS600* potrebno je napajnje od 230 V. Koristi se DC motor od 24 V, pri čemu je nominalna struja 2 A. Maksimalna težina kapije kojom se može upravljati je 600 kg [2].

Slika 1.1: Izgled kućišta modela *iS600* (slika preuzeta iz [1])



Slika 1.2: Dimenzije kućišta modela *iS600* (slika je preuzeta iz [2])

Pored svega navedenog postoji mogućnost korištenja baterije u slučaju prekida napajanja. Baterije se kreću od 2.6 Ah pa sve do 15 Ah. Postoji i opcija korištenja solarnih panela za napajanje uređaja. Snaga solarnih panela se kreće od 5 W do 60 W [1].

Prednost kod modela *iS600* je u korištenju RF predajnika *PentaFOB*. Navedeni predajnik posjeduje mogućnost odašiljanja signala na 5 različitih kanala. Predajnik koristi koristi FHSS (end. *frequency hopping spread spectrum*). Na taj način postoji veoma mala vjerovatnoća da dođe do interferencije signala. Frekvencije na kojima predajnik odašilje signal se kreće u rasponu od 433.1 MHz do 434.7 MHz

Iz svega navedenog može se primijetiti da postoji više proizvođača koji imaju svoja rješenja za posmatrani problem. Njihova rješenja u suštini se ne razlikuju značajno. U zavisnosti od potrebe postoje određene varijacije kako bi se navedena rješenja mogla iskoristiti u više slučajeva. Mogu se primijetiti razni načini komunikacije sa upravljačkim modulom. Moguća je komunikacija pomoću RF primopredajnika, Wi-Fi modula, bluetooth-a i slično. Najčešće korišteni način komunikacije je pomoću RF primopredajnika. U daljoj obradi teme model *iS600* će se uzeti kao referentni model.

1.3 Prijedlog rješenja

Na osnovu prethodno obrađenih modela može se specificirati željeno rješenje uređaja za upravljanje kapijom. Uređaj će sadržavati: upravljački modul, komponente za komunikaciju i upravljanje, senzore za sigurnost i slično. Postoji potreba da se definiše broj potrebnih pinova za realizaciju rješenja. Modul će koristiti sljedeće pinove:

- 3 pina (2 PWM pina) za motor i drajver,
- 2 pina za RF prijemnik i predajnik,
- 2 pina za 2 PIR senzora,
- 2 pina za LCD,
- 8 pinova za keypad,
- 1 pin za svjetlosnu/zvučnu indikaciju,
- 2 pina za senzore udaljenosti,
- 2 pina za optički prijemnik i predajnik.

Ukupan broj potrebnih pinova je 22 (od čega 2 pina trebaju imati PWM funkcionalnost, 2 pina za I2C komunikaciju i 4 pina za UART komunikaciju). Na osnovu ovog podatka donesena je odluka da se koristi ESP32 modul [9]. Navedeni modul ima dovoljan broj I/O pinova i pored toga već ima ugrađene dvije komponente za komunikaciju: Wi-Fi modul i bluetooth modul.

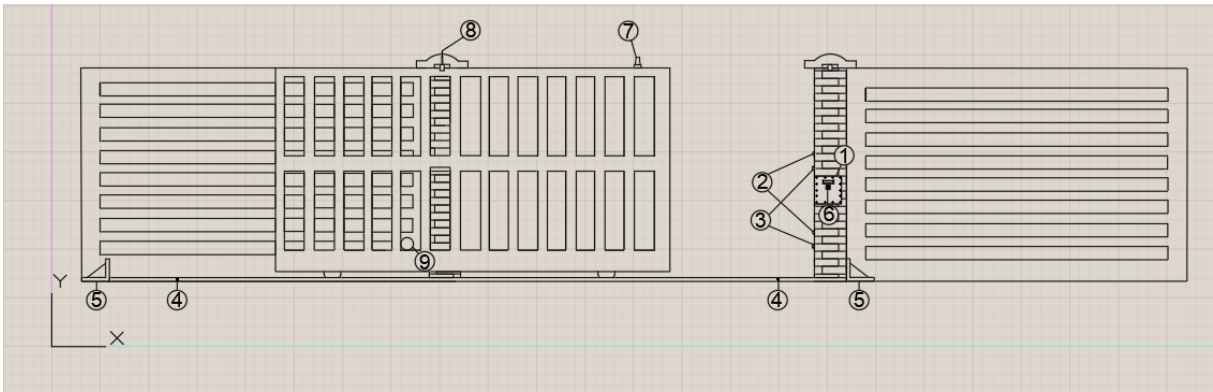
Poglavlje 2

Hardverske komponente upravljačkog modula

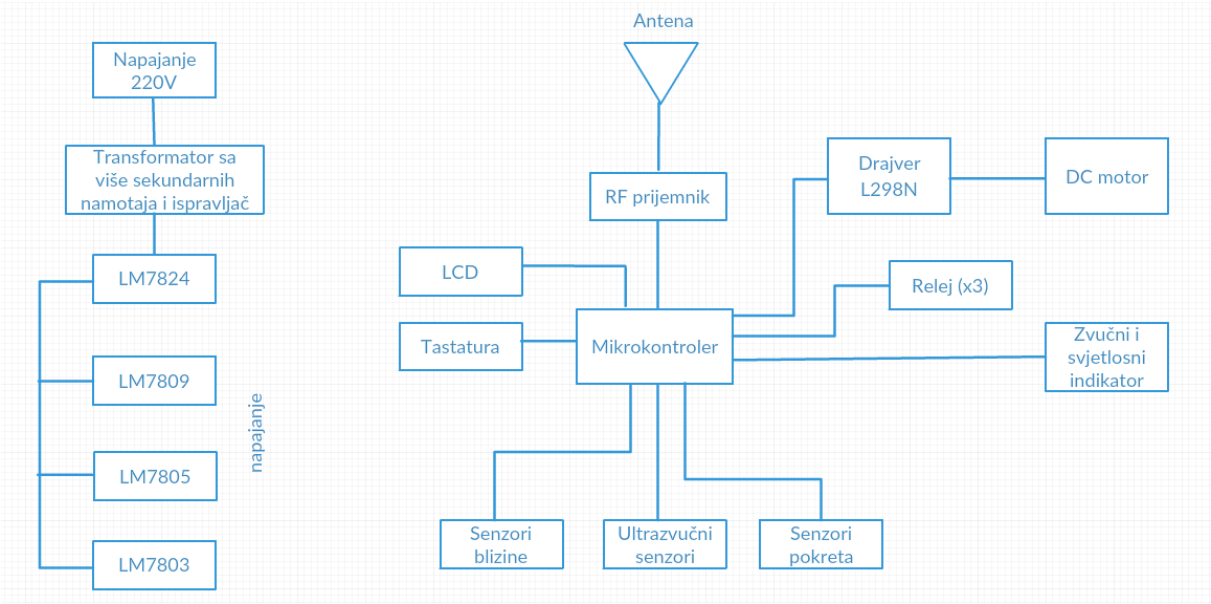
2.1 Opis hardverske implementacije rješenja

Na slici 2.1 može se vidjeti skica kapije. Cilj rada je da se napravi uređaj pomoću kojeg postoji mogućnost upravljanja kapijom sa određene udaljenosti. Uređaj će se moći koristiti na već postojećim kapijama bez potrebe za novom konstrukcijom. Planirana je i konstrukcija minijaturnog modela koji će služiti za prikaz upravljanja kapijom. Prikazani elementi na slici 2.1 su:

1. Upravljačka kutija
2. Senzori pokreta
3. Ultrazvučni senzori
4. Krajnji prekidači
5. Mehanički graničnici
6. LCD sa tastaturom
7. Svjetlosni i zvučni indikator
8. Antena
9. Motor



Slika 2.1: Skica kapije



Slika 2.2: Blok dijagram sistema

2.2 Komunikacija

Prilikom uspostavljanja komunikacije između dvije tačke često se javi potreba za prijenosom informacija na velike udaljenosti. Zbog toga nastaju razni problemi, od kojih je vjerovatno najvažniji izobličavanje signala uslijed djelovanja šuma. Neki od izvora šuma su elektromagnentni talasi nastali u prijenosnim vodovima i električnim mašinama. Signali se mogu prenositi električnim, pneumatskim, optičkim putem ili radio vezom u analognom ili digitalnom obliku.

Električni prijenos podataka je najjednostavniji način prijenosa. Međutim, ovaj način prijenosa je otežan zbog djelovanja šuma.

Značajan razvoj optičkih komunikacija je doveo do toga da se optički sistemi za prijenos podataka sve češće koriste. Prednosti optičkih komunikacija se ogledaju u tome što su šum i interferencija skoro u potpunosti eliminisani. Prijenos se obično vrši kroz fiber-optičke kablove.

Radio veza se obično koristi za prijenos informacija na udaljenosti do 650 km. Posebnim tehnikama moguća je realizacija radio veza i za komunikaciju u svemirskim prostranstvima. Pored navedenog, radio veze se često koriste i na malim udaljenostima zbog toga što je otežano uspostavljanje drugih načina komunikacije [14].

Prednost elektroničke strukture za daljinsko upravljanje kapijom je u tome što omogućava upravljanje kapijom sa udaljenosti, pri čemu vlasnik nema potrebe da izlazi i ručno otvara kapiju niti da gubi vrijeme na tu aktivnost. Primarni cilj je da se omogući komunikacija sa upravljačkim modulom sa velike udaljenosti. Postoje razne varijante komunikacije, a posebna pažnja će se obratiti na:

- Wi-Fi,
- RF,
- Bluetooth.

2.2.1 Wi-Fi

Prisutan na radnom mjestu, kod kuće, u kafićima, na aerodromima, bežični LAN (eng. *Local Area Network*) je postao jedan od najvažnijih tehnologija za pristup mreži. Iako je postojalo mnogo standarda (razvijenih 1990-ih godina) IEEE komunikacijskih standard 802.11 je standard koji se pokazao kao najbolji (ovaj standard je poznat kao Wi-Fi). Postoji više vrsta 802.11 standarda koji se razlikuju po opsegu na kojem rade i brzinama koje se mogu postići [15].

Standard	*Frekvencijski opseg*	*Brzina prijenosa podataka*
802.11b	2.4GHz-2.485GHz	do 11Mbps
802.11a	5.1GHz-5.8GHz	do 54Mbps
802.11g	2.4GHz-2.485GHz	do 54Mbps

Tabela 2.1: Poređenje IEEE 802.11 standarda

Wi-Fi komunikacija je postala veoma široko rasprostranjena zbog prednosti koje nudi:

- mobilnost,
- efikasnost,
- mala cijena instalacije [16].

Nasuprot navedenom postoje i određeni nedostaci. Ovi nedostaci su razlog zašto žičane komunikacije (*Ethernet*) imaju prednost u odnosu na Wi-Fi prilikom izbora načina komuniciranja:

- interferencija,
- brzo opadanje snage signala [15],
- problemi sa vezom,
- sigurnost,
- performanse.

Jedan od nedostataka koji se javlja u bežičnim komunikacijama je nemogućnost uspostavljanja stalne konekcije uređaja sa pristupnom tačkom (eng. *Access Point* daljem tekstu AP). To se dešava iz prostog razloga jer snaga signala AP-a nije veća od praga za detekciju u svim dijelovima prostorije i javljaju se slijepe mrlje. Rješenje navedenog problema je korištenje više pristupnih tačaka [17].

Još jedan važan nedostatak Wi-Fi komunikacije je interferencija. Wi-Fi komunikacija koristi 2.4GHz i 5GHz opsege frekvencije. Opseg 2.4GHz je jedan od ISM (eng. *Industrial, scientific and medical radio bands*) opsega. Problem kod navedenog opsega je u tome što njega već koriste u medicinske svrhe pa postoje primarni korisnici. Drugi korisnici koji žele da komuniciraju u tom opsegu se smatraju sekundarnim korisnicima i ne smiju da ometaju primarne korisnike. Zbog toga se sekundarni korisnici moraju izboriti sa interferencijom od primarnih

korisnika i od drugih sekundarnih korisnika. Navedeni problem se rješava tako što se ograničava dozvoljena izračena snaga svakog uređaja. Drugi opseg na frekvencijama reda 5GHz naziva se U-NII (eng. *Unlicensed National Information Infrastructure band*) opseg. Prednost U-NII opsega je što nema primarnih i sekundarnih korisnika i zbog činjenice da se koriste više frekvencije brzine prenosa informacija su veće. Nedostatak U-NII opsega je što signal brže slabi pa zbog viših frekvencija postoji veće slabljenje signala i samim tim domet je manji [18].

Da bi shvatili važnost zaštite podataka koji se šalju kroz slobodni prostor prvo je potrebno identifikovati osobine bezbjedne komunikacije:

- Povjerljivost. Samo pošiljalac i primalac trebaju da razumiju poslanu poruku.
- Integritet poruke. Poruka dolazi na odredište ista onakva kakva je i poslana.
- Provjera autentičnosti krajnjih tačaka. Pošiljalac mora uvjeriti primaoca da je to stvarno on, isto vrijedi i za primaoca.
- Operaciona bezbjednost. Skoro sve organizacije imaju pristup javnom internetu pa iz tog razloga mogu postati meta napada. U cilju zaštite od navedenog napada koriste se: firewall i sistemi za otkrivanje uljeza (eng. *Intrusion Detection Systems*, IDS).

Komunikacija se vrši pomoću radio signala (komunikacija u slobodnom prostoru) pa je iz tog razloga lakše prisluškivati u odnosu na žičane komunikacije. Javlja se potreba da se izvrši šifriranje poslanih podataka kako bi se spriječio neovlašteni pristup. Uljez potencijalno može da prisluškuje, modificira, ubacuje ili briše poruke ili njihov sadržaj [15]. U cilju zaštite poruka napravljeni su sigurnosni algoritmi. *Wired Equivalent Privacy* (WEP) je primjer takvog algoritma. Napravljen je 1999. godine kao dio IEEE standarda 802.11. u cilju zaštite privatnosti, autentičnosti krajnjih tačaka i integriteta slično kao u žičanim mrežama. Napredniji algoritam zaštite podataka je IEEE 802.11i. On je pružao mnogo sigurniji način enkripcije i širi spektar mehanizama za autentifikaciju. Nakon toga su kreirani *Wi-Fi Protected Access* (WPA) i WPA 2 koji sprječavaju napade bazirane na statističkim analizama. WPA 2 je baziran na *Advanced Encryption Standard* (AES) algoritmu. AES algoritam koristi 128-bitne ključeve pa je mogućnost probijanja zaštite isprobavanjem svih mogućih varijanti šifre značajno smanjena [16].

Na osnovu svega navedenog može se zaključiti zašto je Wi-Fi najčešće korišten način komunikacije u IoT-u. Pojam *Internet of Things* se odnosi na mogućnost povezivanja senzora, aktuatora ili bilo kakvih uređaja na internet. Na taj način se može značajno doprinijeti kvalitetu života. Napretkom mreža i velike raznolikosti postojećih uređaja javila se potreba za sinhronizacijom, odnosno definisanjem standarda. *Internet Protocol* (IP) se pokazao kao najbolji kandidat za postizanje sinhronizacije. *Internet Protocol for Smart Objects* (IPSO) alijansa je prepoznala prednosti koje nudi IP. Ovaj pristup je omogućio širok spektar primjene uređaja u industriji, prevozu, u privatne svrhe i slično. Do sada su se uglavnom koristili ZigBee i drugi IEEE 802.15.4 protokoli za mrežu senzora zbog energetske efikasnosti. Međutim, nedavnim razvojem energetski efikasnih Wi-Fi komponenata one su se počele nametati kao poželjno rješenje. Njihova prednost je u mogućnosti korištenja već postojećih infrastruktura sa ugrađenim IP mrežama. To je razlog zbog kojeg se ostvaruje ušteda i brža implementacija.

2.2.2 RF

RF 433MHz moduli se često koriste u sistemima na daljinsko upravljanje. Primjeri njihove upotrebe se mogu naći kod dronova, robota, industrijskog upravljanja i slično. RF moduli se sastoje od prijemnika i predajnika. *Radio frequency* (RF) opseg obuhvata područje frekvencija od 3kHz do 300GHz. Iako radio frekvencija označava učestalost oscilacija, pojam *radio frekvencije* i njegova skraćenica RF se koriste kao sinonim za radio komunikaciju. Pomoću navedenih pojmova se opisuju bežične komunikacije.

Antene se koriste za prijem signala i za povećanje dometa. Prilikom prijema antena će detektovati na hiljade signala. Iz tog razloga je potrebno koristiti i radio tuner zbog izdvajanja frekvencija od interesa. Jednostavna realizacija tunera može se postići pomoć rezonatora koji se sastoji od kondenzatora i zavojnice. Rezonator pojačava oscilacije unutar određenog opsega frekvencija dok istovremeno slabi oscilacije van opsega. Udaljenost na kojoj se može koristiti radio signal obično zavisi od predajne snage, kvaliteta prijemnika, tipa, veličine i visine antene, modulacije prenosa, šuma i interferirajućih signala. Frekvencija koja se koristi za prenos ne utječe značajno na prenos kao navedeni parametri.

RF modul radi na radio frekvencijama. U RF sistemima digitalni podaci su predstavljeni pomoću različitih amplituda nosioca. Ova modulacija je poznata kao amplitudska modulacija (ASK, eng. *Amplitude Shift Keying*). Prednosti RF-a u odnosu na IR (infracrveni) ospeg su:

- RF signal ima veći domet u odnosu na IR.
- IR signali se najčešće koriste u LOS (eng. *line-of-sight*) komunikacijama dok prijemnici RF signala mogu detektovati signal iako postoji prepreka između prijemnika i predajnika.
- RF komunikacije su mnogo pouzdanije u odnosu na IR komunikacije iz razloga što RF komunikacije koriste određenu frekvenciju za razliku od IR-a [19].

2.2.2.1 ASK modulacija

ASK je pojam koji se odnosi na modulaciju signala nosioca sa diskretnim amplitudama koje jednoznačno označavaju simbole ili bite. U slučaju kada se amplitudska modulacija predstavlja sa

$$\alpha_m = 2m : m = 0, 1 \tag{2.1}$$

i kada je primijenjena na jednu kvadraturnu komponentu noseće frekvencije onda se naziva OOK modulacija (eng. *On-off Keying*). Ukoliko se predstavlja na sljedeći način

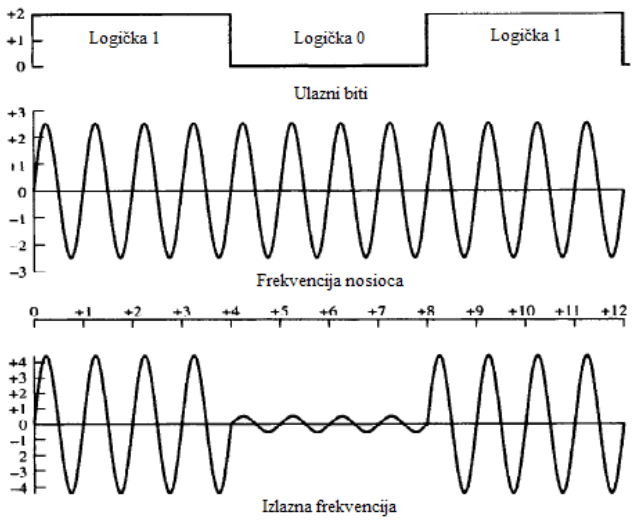
$$\alpha_m = -1 + 2m : m = 0, 1 \tag{2.2}$$

onda je to antipodna ASK modulacija, koja u slučaju da se primijeni na RF nosioc ima iste osobine kao i BPSK modulacija (eng. *Binary Phase Shift Keying*). Opštiji slučaj primjene ASK modulacije samo na jednu kvadraturnu komponentu je za

$$\alpha_m = -(M-1) + 2m : m = 0, 1, ..., M-1 \tag{2.3}$$

i u tom slučaju se naziva impulsno amplitudska modulacija (PAM, eng. *Pulse Amplitude Modulation*) ili još opštije M-arni PAM. Još jedna važna modulacija koja se koristi, a koja je povezana sa navedenim, je kvadraturno amplitudska modulacija (QAM, eng. *Quadrature Amplitude Modulation*). Ona se dobije kada se PAM primijeni na obje kvadraturne komponente.

Izbor određene modulacije zavisi od raznih faktora. Jedan od najvažnijih ciljeva je da se postigne što manja vjerovatnoća greške sa što manjom uloženom snagom (u ovom slučaju se



Slika 2.3: Primjer ASK (OOK) modulacije (slika preuzeta iz [3])

govori o energetskoj efikasnosti). Drugi faktor koji se često uzima u obzir je postizanje što veće brzine sa što manjom širinom spektra (ovdje je riječ o spektralnoj efikasnosti).

Najosnovnija forma ASK modulacije je dobila ime OOK modulacija koja se definiše formulom

$$s_m(t) = A\alpha_m p(t) cos(\omega_c(t)) : |t| \leqslant T/2, \alpha_m = b = \{0,1\} \tag{2.4}$$

gdje je *b* izvor podataka i *p(t)=rect(t/T-mT)* odgovara *p(t)=1* za svako t. Primjer izgleda signala kod ASK (OOK) modulacije se može vidjeti na slici 2.3

Još jedan važan aspekt vezan za performanse modulacija je način detekcije signala. Postoje dva načina detekcije signala:

- koherentna detekcija
- nekoherentna detekcija

Kod koherentne detekcije, ukoliko dođe do greške u fazi, na prijemu se koristi PLL petlja (eng. *Phase Loop Lock*) kako bi se ispravila greška faze. U slučaju nekoherentne detekcije ne koristi se PLL petlja i na taj način se pojednostavljuje konstrukcija prijemnika. Prednost koherentne detekcije je u tome što će postojati manja vjerovatnoća greške ispravnog prijema. Obje varijante imaju svoje prednosti i mane i u zavisnosti od potreba sistema odlučuje se koja će se vrsta detekcije koristiti [20].

2.2.2.2 Protokoli definisani u *"Rc-switch.h"* biblioteci za ASK modulaciju

Biblioteka *"Rc-switch.h"* je namijenjena za korištenje RF 315/433 MHz primopredajnika sa Raspberry Pi modelima, ESP8266 i ESP32 čipovima. Unutar biblioteke definisano je sedam različitih protokola koji se mogu koristiti. Svaki protokol se sastoji od četiri komponente (kombinacijom navedenih komponenti protokola dobijaju se tri-state biti, koji se također šalju prilikom komunikacije) [21]:

- dužine impulsa - predstavlja vremensko trajanje impulsa u mikrosekundama,
- bita za sinhronizaciju - neka je definisan par $\{1, 31\}$. To znači da se niz sastoji od jedne logičke "1" i trideset i jedne logičke "0", pa je ukupan broj bita u nizu trideset i dva [22],

- logičkih ”1”,
- logičkih ”0”.

Program 2.1: Protokoli definisani u *”Rc-switch.h”* biblioteci

```
#if defined(ESP8266) || defined(ESP32)
static const RCSwitch::Protocol proto[] = {
#else
static const RCSwitch::Protocol PROGMEM proto[] = {
#endif
  { 350, {  1, 31 }, {  1,  3 }, {  3,  1 }, false },
  { 650, {  1, 10 }, {  1,  2 }, {  2,  1 }, false },
  { 100, { 30, 71 }, {  4, 11 }, {  9,  6 }, false },
  { 380, {  1,  6 }, {  1,  3 }, {  3,  1 }, false },
  { 500, {  6, 14 }, {  1,  2 }, {  2,  1 }, false },
  { 450, { 23,  1 }, {  1,  2 }, {  2,  1 }, true  },
  { 150, {  2, 62 }, {  1,  6 }, {  6,  1 }, false }
};
```

Razlika se može primijetiti za svaki protokol u načinu kako su definisane njihove vrijednosti. Vidi se razlika u trajanju impulsa, dužini niza bita za sinhronizaciju i u broju logičkih ”0” i ”1”. Ukoliko nijedan protokol ne odgovara za određeni prijemnik ili predajnik, onda se neće moći izvršiti razmjena podataka među njima.

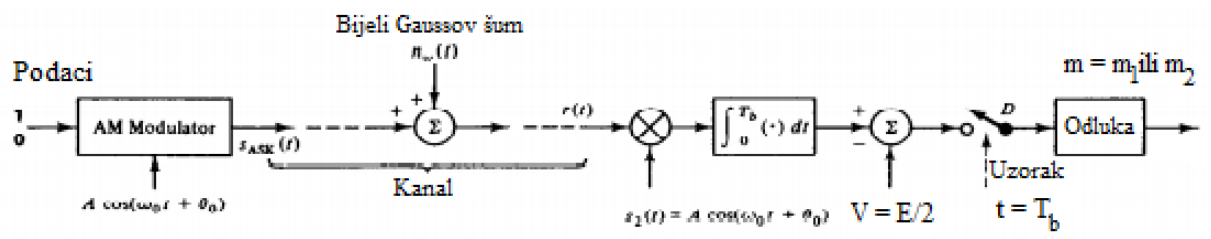
2.2.2.3 Formiranje RF linka podataka

Slanje podataka bežičnim putem je mnogo komplikovanije od korištenja bakarne parice ili optičkih kablova. Bežični prijenos sirovih podataka (’1’ i ’0’) je teško izvodljiv. Zbog toga se prijenos u bežičnim kanalima vrši pomoću moduliranih signala. Prilikom prijenosa signala žičanim kanalima nema velikih smetnji i zbog toga se dobija pouzdan prijenos. U slučaju korištenja bežičnog kanala postoje velika slabljenja prilikom propagacije talasa i ne mogu se koristiti metode za poboljšanje performansi koje se koriste za prijenos žičanim putem (potrebne su određene modifikacije).

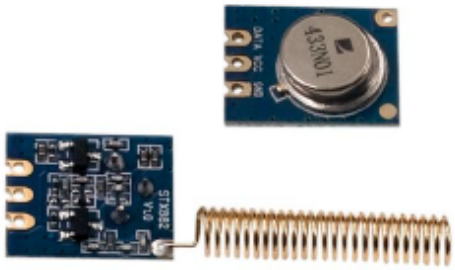
Jedan od najvećih problema kod bežičnh kanala je postojanje interferencije. RF frekvencija koja se obično koristi je 433.93 MHz. Zbog činjenice da se ta frekvencija često koristi i zbog toga što nije potrebna licenca za korištenje iste, postoji mnogo uređaja koji mogu istovremeno raditi na navedenoj frekvenciji. Kao posljedica se javlja interferencija koja utječe negativno na posmatrani signal. Smatra se dobrim rezultatom ukoliko je moguće koristiti navedenu frekvenciju za 95% slučajeva. Obično su očekivane vrijednosti manje. Kod žičanih medija pouzdanost od približno 100% se dobija zbog toga što je medij zaštićen. Međutim, skoro je nemoguće zaštiti bežični medij. Jedan od načina zaštite od interferencije u bežičnim kanalima je da se signal šalje u intervalima umjesto konstantog signala.

Kako bi se postigla dodatna zaštita signala i poboljšao prijenos dodaje se redudansa. Na ovaj način dobija se sporiji prijenos informacija, ali pouzdaniji. Jedan od načina dodavanje redudanse je ponovno slanje podataka sa pretpostavkom da smetnja koja je nastala nije dugotrajna. Drugi način je kanalno kodiranje. Pomoću kanalnog kodiranja omogućava se detekcija i/ili korekcija grešaka. Neki od kanalnih kodova koji se koriste u ovu svrhu su FEC (eng. *Forward Error Correction*) i BCH (eng. *Bose-Chaudhuri-Hocquenghem*) kodovi [23].

Prilikom prijenosa podataka ASK (OOK) modulacijom prvo se formira niz bita koji predstavljaju informaciju koja se šalje, nakon čega se vrši modulacija signala. Prilikom prijenosa signala potrebno je modulisati signal kako bi se izvršio prijenos signala kroz medij. Nakon toga



Slika 2.4: Blok dijagram optimalnog koherentnog ASK (OOK) sistema (slika preuzeta iz [4])



Slika 2.5: Fizički izgled STX882 RF predajnika (slika preuzeta iz [5])

signal prolazi kroz medij uslijed čega dolazi do njegovog izobličavanja zbog raznih smetnji koje djeluju na njega (u ovom slučaju bijeli Gaussov šum). Sljedeći korak je prijem signala. Na prijemnoj strani signal prvo prolazi kroz množač, gdje se množi sa istim signalom koji se koristio i u modulatoru. Nakon toga se vrši integriranje signala na svakom simbol intervalu. Poslije integratora signal prolazi kroz sumator, pri čemu se od dobijenog signala na izlazu iz integratora oduzima referentna vrijednost. Zatim se vrši sempliranje signala i odlučivanje na osnovu vrijednosti koja se dobije nakon sumatora. Na slici 2.4 može se vidjeti blok dijagram ASK sistema [4].

2.2.2.4 RF predajnik STX882

STX882 je ASK predajnik malih dimenzija i velike snage. Navedeni predajnik će se koristiti u svrhu testiranja SRX882 prijemnika. Njegove specifikacije se mogu vidjeti u tabeli 2.2 [5].

Parametar	*Min.*	*Tip.*	*Max*	*Jedinica*
Napajanje	1.2	3.0	6	V
Opseg temperatura	-20	25	70	°C
Opseg frekvencija	433.82	433.92	434.02	MHz
Snaga (za napajanje od 3V)	14	15	15.5	dBm

Tabela 2.2: Specifikacije STX882 RF predajnika

Slika 2.6: Fizički izgled SRX882 RF prijemnika (slika preuzeta iz [6])

2.2.2.5 RF prijemnik SRX882

SRX882 je ASK prijemniik koji se često koristi sa STX882 i STX888 predajnicima.Odlikuje ga dobra stabilnost i otpornost an interferenciju. Može raditi u opsegu napona od 2.4 V do 5.5 V, što je prednost za posmatrani problem jer je potrebno napajanje za mikrokontroler 3.3 V. Navedeni prijemnik se može koristiti u mnogim aplikacijama kao što su:

- daljinsko upravljanje vratima,
- bežični sigurnosni alarm,
- bežični prijenos podataka.

Specifikacije prijemnika se mogu vidjeti u tabeli 2.3 [6].

Parametar	*Min.*	*Tip.*	*Max*	*Jedinica*
Napajanje	2.4	5.0	5.5	V
Opseg temperatura	-30		85	°C
Kašnjenje			9	ms
Opseg frekvencija	433.82	433.92	434.02	MHz
Osjetljivost		-110	-107	dBm
Širina opsega		200		khz

Tabela 2.3: Specifikacije SRX882 RF prijemnika

2.2.3 Bluetooth

Smanjenjem cijene i potrošnje radio uređaja pojavila se mogućnost da se naprave elektronički uređaji koji će se koristiti u svrhu kreiranja pametnih kuća, senzorskih mreža i sličnih aplikacija. Dva uređaja su se pokazala kao dobra rješenja: Bluetooth i Zigbee.

Bluetooth omogućava uspostavljanje veze između bežičnih uređaja na manjim udaljenostima. Primopredajnik se nalazi umjesto konektorskog kabla u uređajima kao što su mobilni telefoni, laptopi, projektori i slično. Bluetooth se najčešće koristi za komunikacije na manjim udaljenostima reda 10 m (za snage predajnika reda 1 mW). Može se koristiti za komunikaciju između laptopa i projektora. Domet se može povećati do reda 100 m ukoliko se predajna snaga poveća na 100 mW. Bluetooth radi na frekvenciji od 2.4 GHz zbog toga što je to nelicencirani opseg. Kako radi na nelicenciranom opsegu postoji opasnost od pojave interferencije. Kako bi se izbjegla interferencija ograničava se snaga predajnika.

Bluetooth standard omogućava jedan asinhroni podatkovni kanal sa brzinom od 723.2 Kbps. U ovom načinu rada, poznatom kao ACL (eng. *Asynchronous Connection-Less*), postoji još jedan povratni kanal sa brzinom od 57.6 Kbps. Drugi način rada dozvoljava tri sinhrona kanala sa brzinama od 64 Kbps. Ovaj način rada se naziva SCO (eng. *Synchronous Connection Oriented*) i najčešće se koristi za prijenos govora.

Rutiranje asinhronih podataka se izvršava pomoću protokola koji je baziran na frekvencijskom skakanju signala sa brzinama od 1600 skokova u sekundi, ova tehnika je poznata kao FHSS (eng. *Frequency Hopping Spread Spectrum*) tehnika. Bluetooth koristi frekvencijsko skakanje za višekorisnički pristup, pri čemu je razmak između nosioca 1 MHz. Obično se koristi do 80 različitih frekvencija sa ukupnom širinom opsega od 80 MHz. Doći će do interferencije ukoliko različiti uređaji u istom trenutku koriste istu frekvenciju. Porastom broja uređaja raste i vjerovatnoća pojave interferencije. U razvoju Bluetooth standarda učestvovale su mnoge velike kompanije (Ericsson, Intel, Microsoft...) i koristi se u mnogim uređajima kao što su slušalice za mobilne telefone, bežični USB, RS232 konektori i slično.

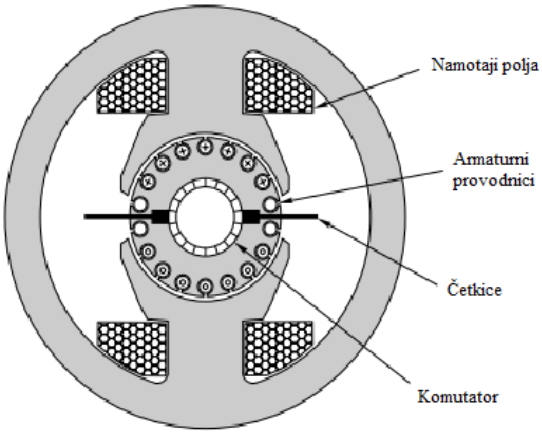
Poželjno je još spomenuti i Zigbee uređaj koji je dizajniran tako da ima još manju potrošnju i cijenu od Bluetooth uređaja. Zigbee koristi IEEE 802.15.4 standard. Radi u istom ISM opsegu kao i Bluetooth i može se povezati sa 255 uređaja po mreži. Podržava brzinu prijenosa podataka do 250 Kbps sa dometom do 30 m. Brzina prijenosa je manja od one koju omogućava Bluetooth uređaj. Cilj korištenja Zigbee uređaja je rad po nekoliko mjeseci ili godina bez potrebe za punjenjem istog [18].

2.3 DC motor i L298N drajver za motor

2.3.1 DC motor

Primjena DC motora je veoma široka. DC motori se koriste u industriji, robotici i slično. Opseg snage na kojoj rade motori je širok. Kreće se od nekoliko vati do reda megavati. Svi motori, bez obzira na snagu, imaju istu strukturu (koja se može vidjeti na slici 2.7). Motor se sastoji od dva odvojena kola. Manji par terminala je povezan sa namotajima polja, koji okružuju oba pola i obično su serijski vezani. Sva ulazna snaga se pretvara u toplotnu energiju. Glavni terminali prenose ”radnu” struju na četkice koje prave klizni kontakt sa namotajima armature na rotoru [7].

Moguće je dizajnirati DC motor na bilo koji napon napajanja, ali iz nekoliko razloga rijetko se mogu naći motori ispod 6 V ili puno veći od 700 V. Mali DC motori do 100 W imaju brzinu oko 12 000 RPM (eng. *Rounds Per Minute*), a većina srednjih i velikih motora imaju brzine



Slika 2.7: Struktura DC motora (slika preuzeta iz [7])

reda 3 000 RPM. Danas se često mogu naći motori sa standardizovanim vrijednostima (110 V, 220-240 V ili 380-440 V na 50 ili 60 Hz). Potrebno je spomenuti da se mogu javiti problemi sa četkicama i komutatorima pri velikim brzinama.

Kontrolni sistem DC motora se sastoji od dva dijela: motora koji pretvara električnu energiju u mehaničku koja se koristi za upravljanje teretom i kontrolera koji kontrolom električne energije upravlja motorom. Moguće je upravljanje brzinom motora i obrtnom silom pomoću preciznog upravljanja napona napajanja. Upravljanje se vrši tako što smanjenjem napona se smanjuje brzina motora.

Većina DC motora radi na isti način: struja koja protiče kroz namotaje armature generiše magnetno polje koje uzajamno djeluje sa magnetnim poljem koje je generisano u namotajima polja. Promjenom smjera struje u armaturi (odnosno komutacija) u pravom trenutku uzrokuje da se motor kreće kontinualno [24].

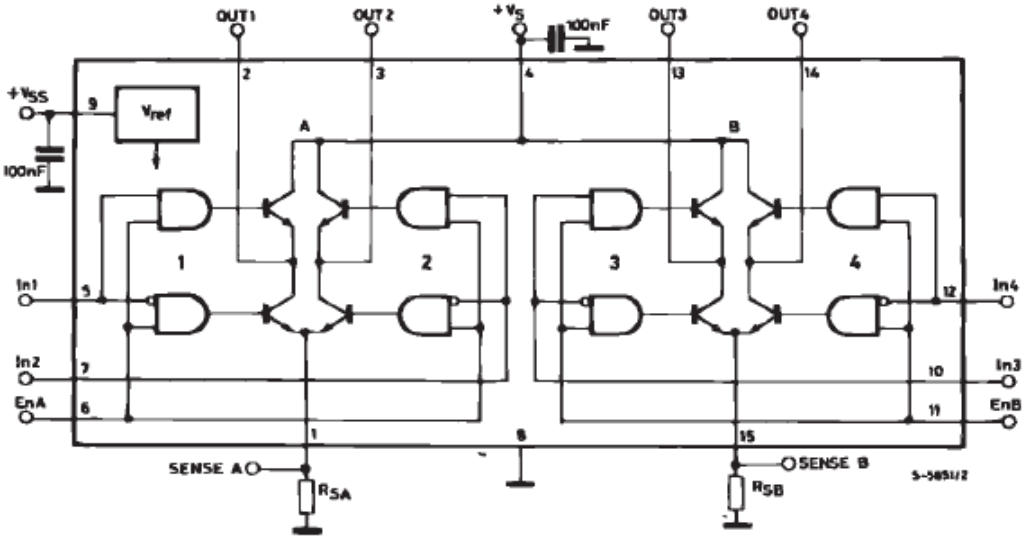
2.3.2 L298N drajver za motor

L298 je integrisano kolo koje se može naći u *Multiwatt* i *PowerSO20* pakovanjima. L298 je drajver koji može raditi na visokim naponima i strujama i dizajniran je da radi sa standardnim TTL logičkim nivoima te da upravlja relejima, DC motorima, solenoidima i step motorima. Prednosti L298 drajvera su što može raditi sa naponima do 46 V, DC strujama do 4 A i ima dobru otpornost na šum.

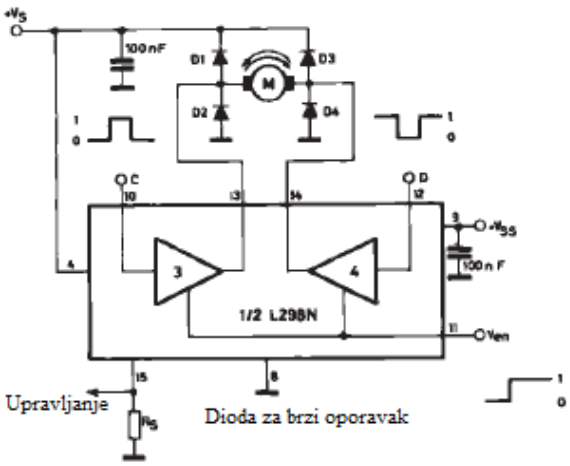
Na komponenti se nalaze dva omogućavajuća ulaza koji služe za aktivaciju i deaktivaciju uređaja koji su povezani na L298, neovisno o ulaznim signalima. Na drajveru se nalazi dodatni ulaz za napajanje u slučaju potrebe za većim napajanjem uređaja koji je povezan na drajver, kako bi napon napajanja za logičke nivoe ostao pri nižim vrijednostima. L298 dolazi u tri različite varijante u zavisnosti od pakovanja u kojem se nalazi:

- L298N - *Multiwatt Vertical*
- L298HN - *Multiwatt Horizontal*
- L298P - *PowerSO20*

Na slici 2.9 može se vidjeti način upravljanja DC motorom pomoću L298 drajvera. Različite vrijednosti dovedene na ulaze C i D proizvode različite reakcije DC motora [8].



Slika 2.8: L298 blok dijagram (slika preuzeta iz [8])



Ulazi		Funkcije
V_{en} = H	C = H ; D = L	Naprijed
	C = L ; D = H	Nazad
	C = D	Brzo zaustavljanje
V_{en} = L	C = X ; D = X	Zaustavljanje slobodnog kretanja

L = Nizak H = Visok X = Don't care

Slika 2.9: Upravljanje bidirekcionalnim DC motorom (slika preuzeta iz [8])

Slika 2.10: Fizički izgled ESP32-DevKitC razvojne ploče

2.4 Mikrokontroler

Mikrokontroler (MCU, eng. *microcontroller unit*) je računar malih dimenzija implementiran na jednom integrisanom kolu. U modernoj tehnologiji, to su SoC sistemi (Sistem na čipu, eng. *System on a Chip*). Jedna mikrokontrolerska jedinica sadrži jednu ili više procesorskih jezgri, zajedno sa memorijom i programabilnim perifernim ulazno/izlaznim modulima. Programska memorija u obliku feroelektrične RAM, NOR Flash ili OTP ROM memorije je također često prisutna, kao i određena količina RAM memorije. Mikrokontroleri se dizajniraju za ugradbene (eng. *embedded*) aplikacije.

Mikrokontroleri se koriste pri realizaciji proizvoda i uređaja sa automatskim upravljanjem, kao što su upravljački sistemi za pogone automobila, implantacijski medicinski uređaji, daljinski upravljači, uredske mašine, alati, igračke i drugi ugradbeni sistemi. Smanjenjem veličine i cijene u odnosu na dizajn koji koristi diskretne elemente mikroprocesora, memorije i ulazno/izlaznih modula, mikrokontroleri omogućuju ekonomičniji način digitalnog upravljanja mnogo više uređaja i procesa [25].

Uzimajući u obzir potrebe proučavanog rješenja odlučeno je da se koristi čip proizvođača *Espressif*. Odlučeno je za navedenog proizvođača zbog činjenice da njihova rješenja sama od sebe sadrže određene funkcionalnosti koje su potrebne za rješenje posmatranog problema. Proučavani su čipovi u kojima su integrisani Wi-Fi i Bluetooth modul. Navedeni proizvođač nudi sljedeće razvojne ploče:

- ESP32-PICO-KIT
- ESP-WROVER-KIT
- ESP32-DevKitC
- ESP32-LyraTD-MSC
- ESP32-LyraT

Nakon proučavanja navedenih razvojnih ploča odlučeno je da se koristi ESP32-DevKitC zbog jednostavnosti i posjedovanja potrebnih osobina (kao što je broj pinova, Wi-Fi modul, dimenzije razvojne ploče i slično).

2.4.1 ESP32

ESP32 je čip koji posjeduje 2.4 GHz Wi-Fi i Bluetooth koji je dizajnirao TSMC (*Taiwan Semiconductor Manufacturing Company*). Dizajniran je da postigne dobre performanse snage i RF performanse, pri čemu pokazuje robusnost, prilagodljivost i pouzdanost u mnogim primjenama.

Dizajniran je za IoT (eng. *Internet of Things)* aplikacije. Posjeduje osobine čipova male potrošnje, više načina rada i dinamično skaliranje snage. Izlazna snaga pojačavača snage je prilagodljiva u zavisnosti od potreba. Teži se ka tome da se postigne najbolji odnos između dometa komunikacije, brzine prenosa i potrošnje snage.

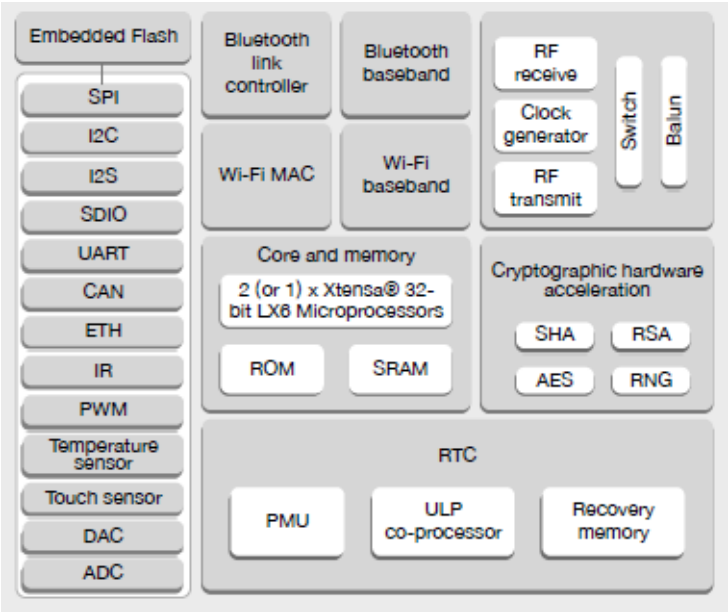
Navedeni modul je veoma dobro rješenje za IoT primjenu gdje se koristi Wi-Fi i/ili Bluetooth, pri čemu posjeduje oko 20 eksternih komponenti (kao što su pojačavač snage, malošumni pojačavač, filtri i slično). Jedna od prednosti ESP32 je u tome što zauzima poprilično malu površinu na PCB-u (eng. *Printed Circuit Board*). Neke od potencijalnih primjena posmatranog čipa su:

- igračke koje se upravljaju pomoću Wi-Fi,
- pametne kuće,
- uređaji za praćenje zdravlja,
- monitori za bebe,
- OTT (eng. *Over The Top*) uređaji.

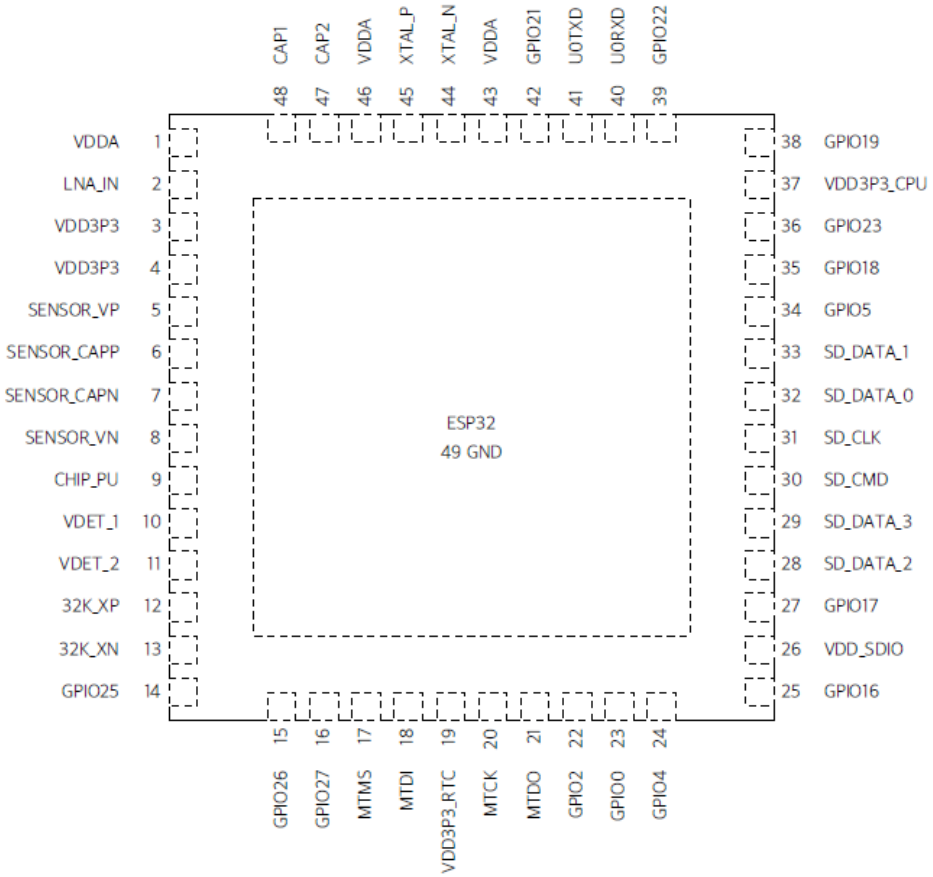
2.4.1.1 Funkcionalni moduli, mogućnosti i raspored pinova ESP32 čipa

ESP32 posjeduje sljedeće karakteristike:

- Xtensa single/dual-core 32-bitni LX6 mikroprocesor, do 600 DMIPS (eng. *Dhrystone Million Instructions per Second*),
- 448 kB ROM,
- 520 kB SRAM,
- napajanje od 2.3 V do 3.6 V,
- Real Time brojač (RTC, eng. *Real Time Counter*),
- 34 programabilna GPIO (eng. *General-purpose input/output*),
- 4 x SPI, 2 x I2S, 2 x I2c, 3 x UART,
- kriptografska zaštita pomoću AES, Hash (SHA-2), RSA, ECC, RNG.



Slika 2.11: ESP32 blok dijagram (slika preuzeta iz [9])



Slika 2.12: Raspored pinova ESP32 čipa (slika preuzeta iz [9])

2.4.1.2 ESP32 Wi-Fi karakteristike

ESP32 koristi TCP/IP i 802.11 b/g/n WLAN MAC protokol. Podržava set baznih usluga (BSS, eng. *Basic Service Set*) STA i SoftAP operacije unutar distribuirane kontrolne funkcije (DCF, eng. *Distributed Control Function*). Upravljanje potrošnjom se vrši minimalnom interakcijom sa hostom kako bi se minimizirao period aktivnosti.

ESP32 Wi-Fi podržava sljedeće funkcije:

- 802.11b i 802.11g brzine prenosa podataka,
- 802.11n MCS0-7 u obje širine opsega 20 MHz i 40 MHz,
- do 150 Mbps brzina prenosa podataka,
- do 20.5 dBm predajna snaga,
- prilagodljiva predajna snaga,
- 4 x Wi-Fi interfejsa,
- defragmentacija,
- antenski *diversity*.

Proučavani čip podržava antenski *diversity* pomoću eksternog RF prekidača. Jedan GPIO upravlja RF prekidačem i bira antenu sa najboljim signalom kako bi se minimizirao efekat kanalnog *fading*-a. ESP32 posjeduje i konfigurabilnu arbitražu paketskog saobraćaja (PTA, eng. *Packet Traffic Arbitration*) koja omogućava fleksibilnu i pravovremenu potporu istovremenog korištenja Wi-Fi i Bluetooth-a. PTA je kombinacija FDM-a (eng. *Frequency Division Multiplexing*) i TDM-a (eng. *Time Division Multiplexing*) [9].

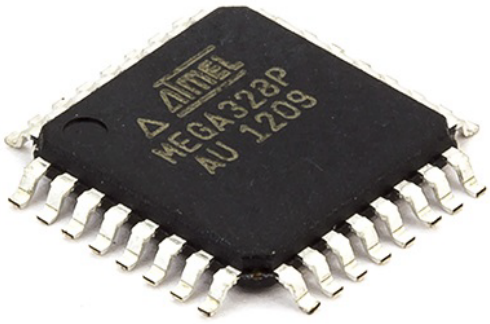
2.4.2 Arduino

2.4.2.1 Atmel ATmega328p

Atmel ATmega328P je 8-bitni CMOS mikrokontroler male snage baziran na tzv. AVR poboljšanoj (eng. *enhanced*) RISC arhitekturi. Fizički izgled mikrokontrolera ATmega328P može se vidjeti na slici 2.13.

Atmel AVR jezgra kombinuje bogat instrukcijski set sa 32 registra opće namjene. Svaki od ta 32 registra je direktno spojen na aritmetičko-logičku jedinicu (ALU), što omogućuje pristup ka dva nezavisna registra pomoću jedne instrukcije koja se izvršava u jednom takt ciklusu. Rezultujuća arhitektura je efikasnija u smislu programiranja, te postiže procesorske brzine blizu 1 MIPS (milion instrukcija u sekundi) pri radnoj frekvenciji od 1 MHz, što je i do deset puta brže od konvencionalnih CISC mikrokontrolera. Ova činjenica omogućuje projektantima sistema da optimiziraju dizajn uređaja u smislu potrošnje, odnosno, u smislu brzine procesiranja. Mikrokontroler ATmega328P se proizvodi u četiri fizički različite varijante (pakovanja):

- 28-pin PDIP,
- 28-pin MLF,
- 32-pin TQFP,
- 32-pin MLF.



Slika 2.13: Fizički izgled mikrokontrolera ATmega328P, 32-pin TQFP (slika preuzeta iz [10])

2.4.2.2 Funkcionalni moduli i mogućnosti mikrokontrolera ATmega328P

Mikrokontroler ATmega328P posjeduje sljedeće funkcionalnosti:

- 32KB sistemski programabilne Flash memorije sa Čitaj-Dok-Pišeš (eng. *Read-While-Write)* sposobnostima,
- 1KB EEPROM memorije,
- 23 I/O linije opće namjene,
- Real Time brojač (RTC),
- 1 bajtno-orjentirani 2-žični serijski interfejs (I2C),
- 8-kanalni 10-bitni AD konvertor,
- SPI serijski port,
- 6 softverski selektabilnih modova uštede energije.

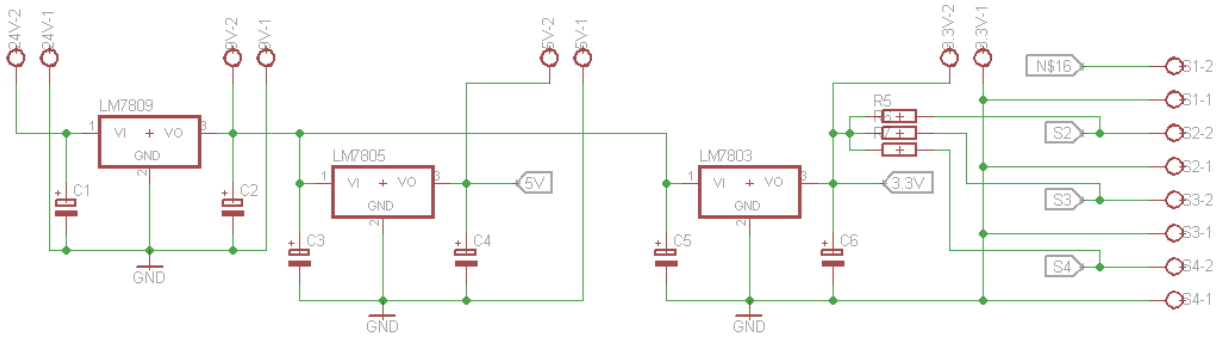
ISP Flash (eng. *In-system Programming*) koji je prisutan na čipu, dozvoljava programskoj memoriji da bude reprogramirana kroz sistem preko SPI serijskog interfejsa, konvencionalnim programatorima neizbrisive (eng. *nonvolatile*) memorije, ali i bootloader programima smještenim na AVR jezgri [10].

2.5 Dizajn štampane pločice - PCB

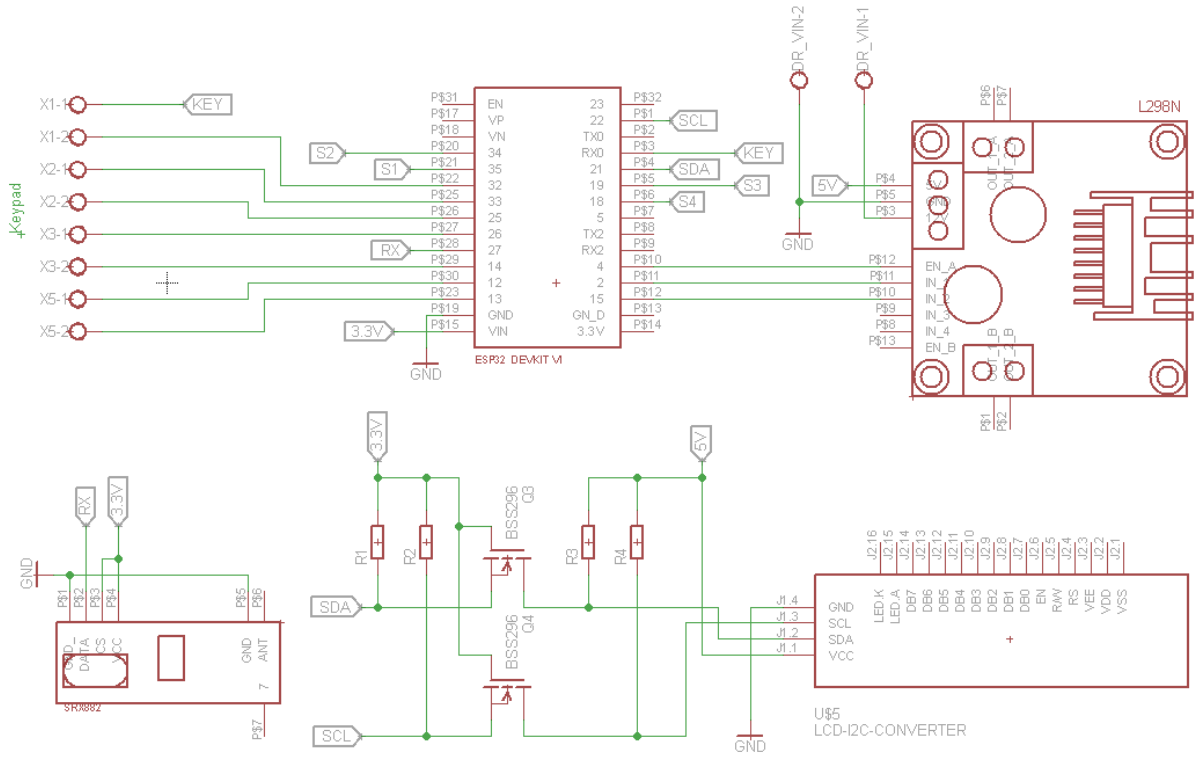
Važan dio realizacije željenog rješenja je dizajn štampane pločice (PCB, eng. *Printed Circuit Board*). Prilikom projektovanja štampane pločice korišten je programski paket *Eagle*.

2.5.1 Izgled štampane pločice i šeme

Na slkama 2.14 i 2.15 prikazana je šema elektroničke strukture za daljinsko upravljanje kapijom dizajnirana u programskom paketu *Eagle*, a na slici 2.16 može se vidjeti izgled štampane ploče u *Eagle*-u.[1]

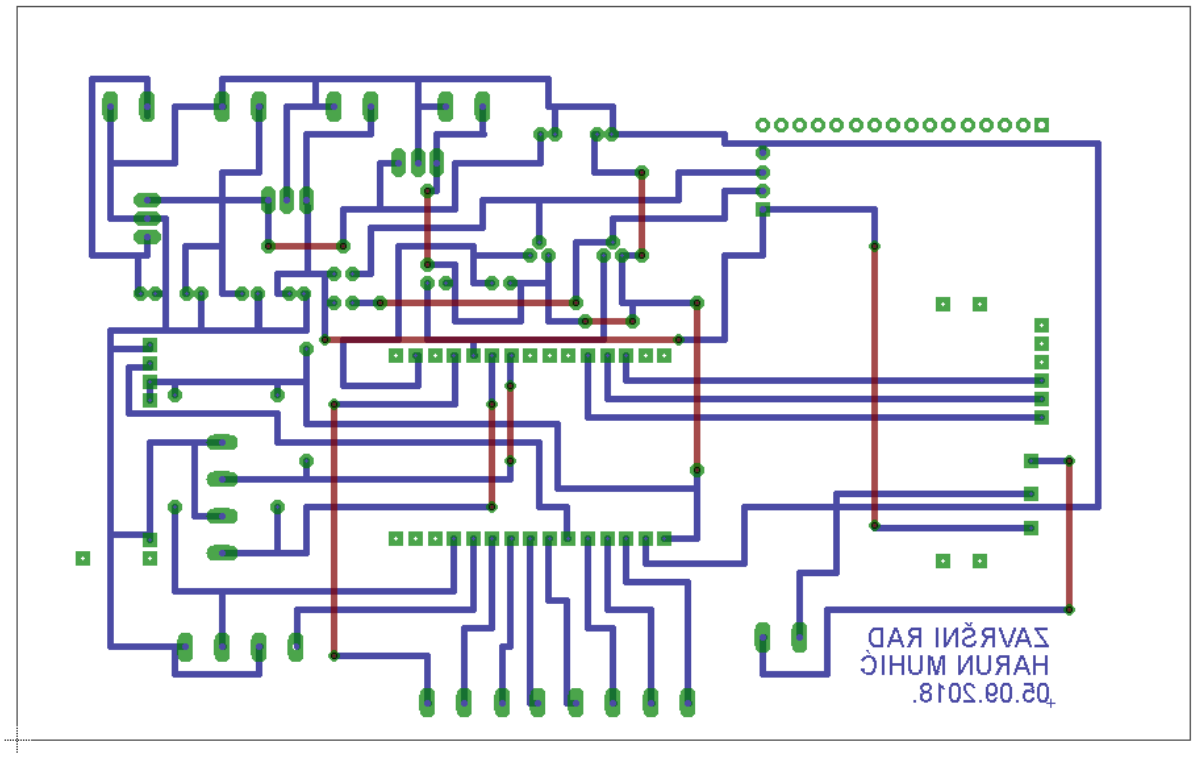


Slika 2.14: Šema elektroničke strukture za daljinsko upravljanje kapijom u *Eagle*-u - dio za napajanje



Slika 2.15: Šema elektroničke strukture za daljinsko upravljanje kapijom u *Eagle*-u - ostale komponente

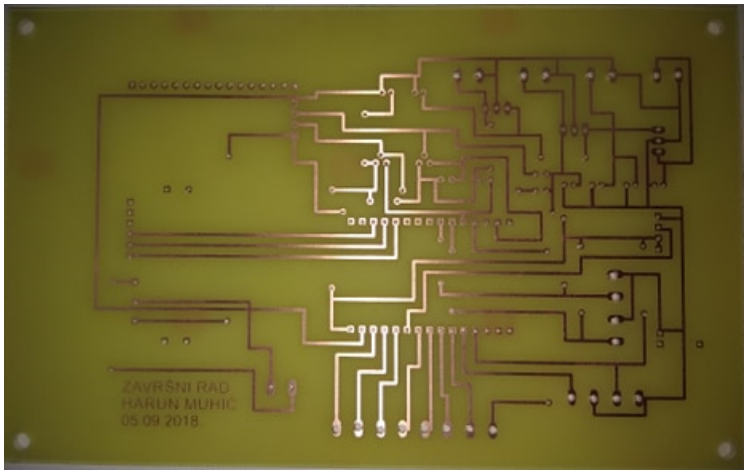
[1]Prilikom pravljenja nove komponente L298N u *Eagle*-u napravljena je greška u dodjeli pinova. Pin koji bi trebao biti za napon od 5 V i pin koji je označen kao 12 V na L298N drajveru su zamijenjeni, pa je greška naknadno ispravljena. Navedena greška se može vidjeti na slici 2.16, ona se nalazi u donjem desnom uglu.



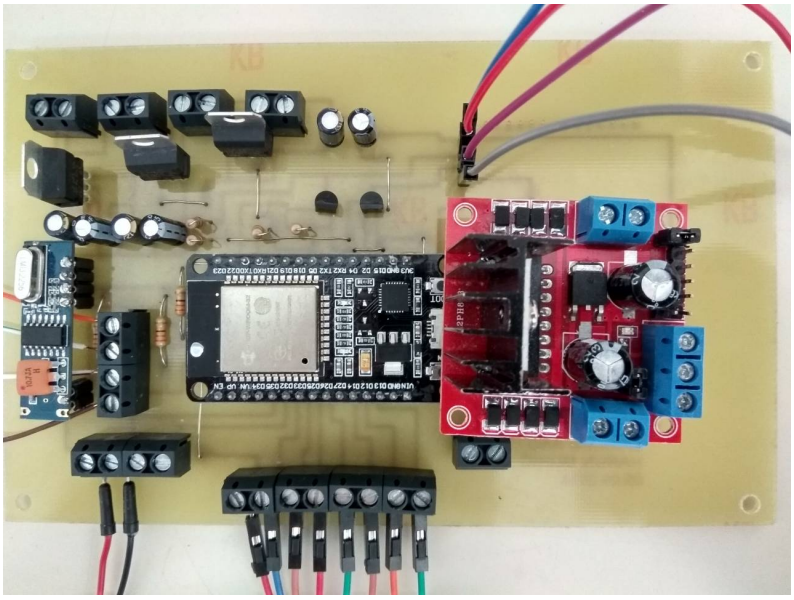
Slika 2.16: Izgled štampane pločice u *Eagle*-u

2.5.2 Fizička realizacija

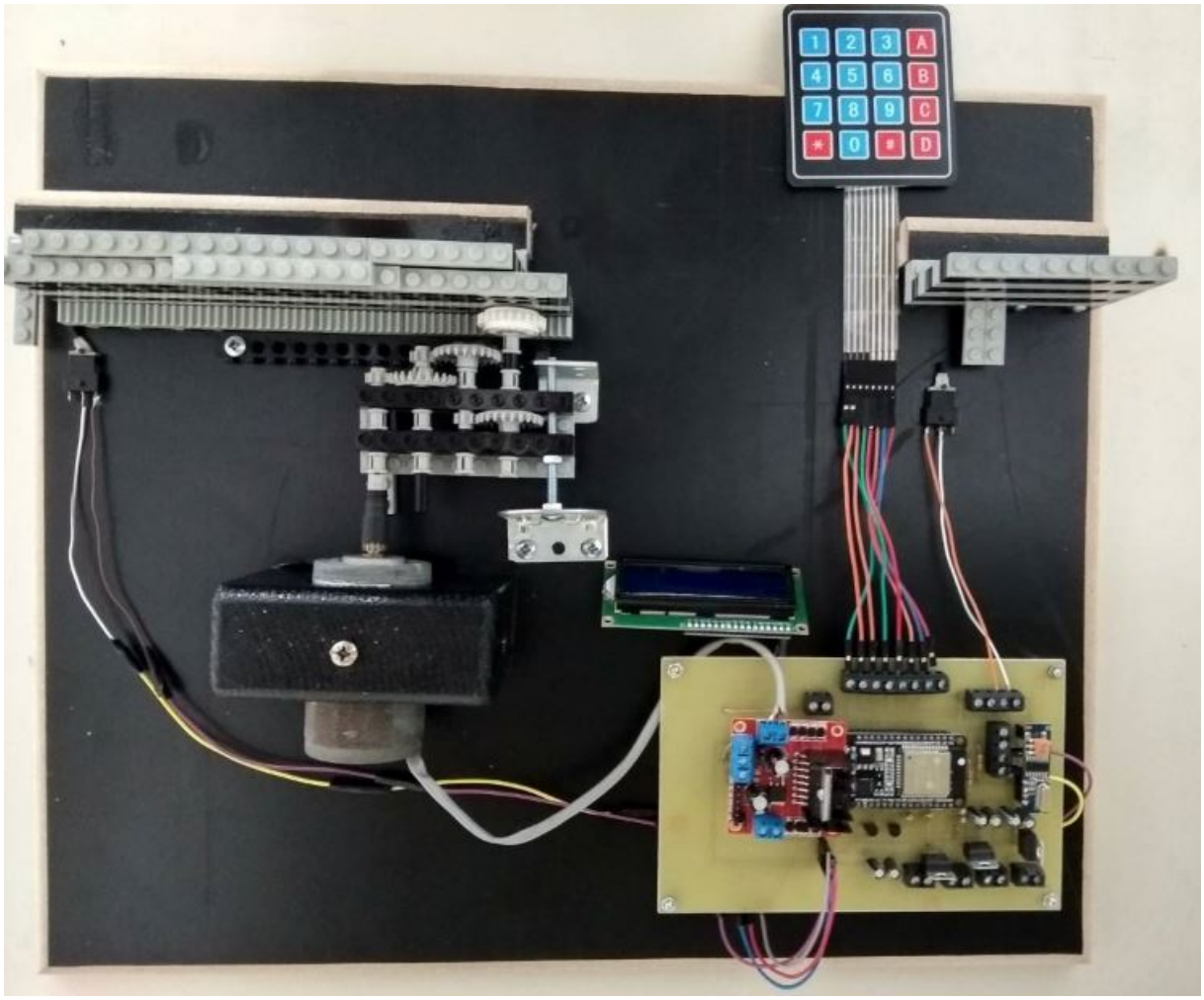
Nakon dizajna pločice u programskom paketu *Eagle* prelazi se na fizičku realizaciju pločice. Na slici 2.17 se može primijetiti izgled pločice prije lemljenja, a na slici 2.18 konačan izgled pločice sa svim korištenim komponentama. Na slici 2.19 može se vidjeti konačan izgled makete.



Slika 2.17: Fizički izgled pločice prije lemljenja - donja strana



Slika 2.18: Fizički izgled pločice nakon lemljenja i postavljanja komponenti



Slika 2.19: Finalni izgled makete

Poglavlje 3

Softverski dio

3.1 Arduino programiranje

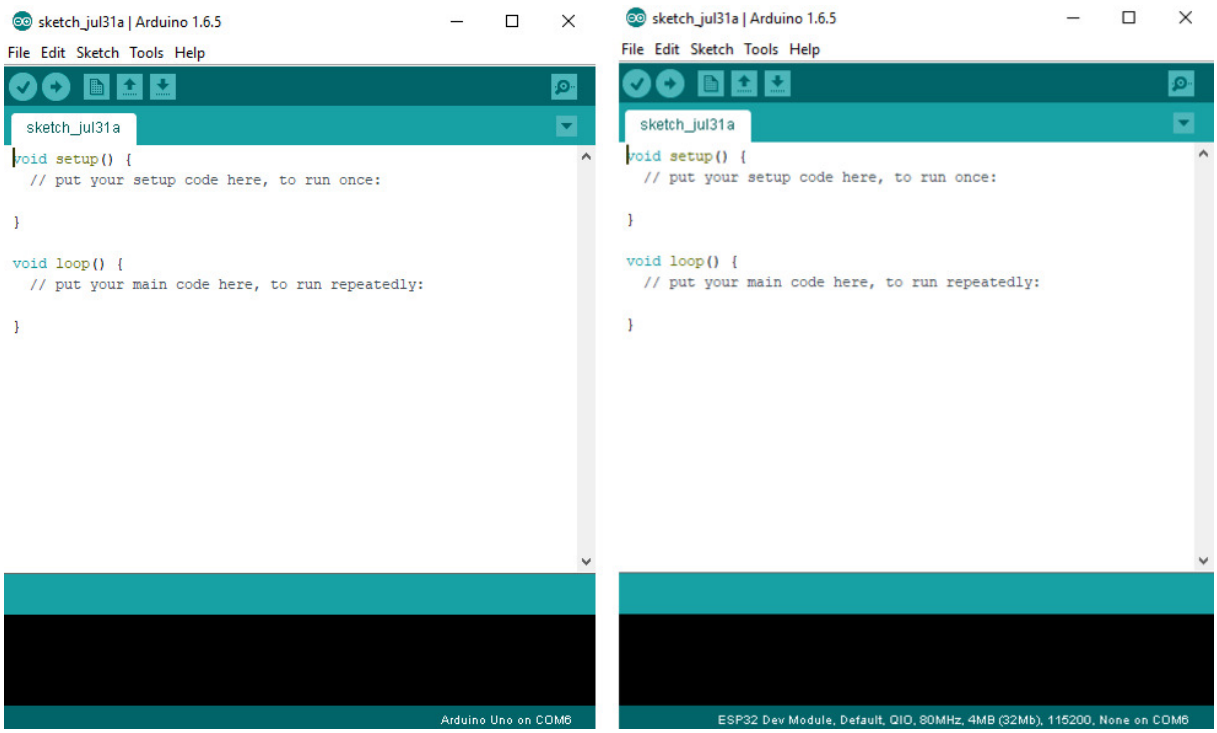
Arduino je fleksibilna hardverska platforma koja se može programirati. Arduino se sastoji od dva dijela: hardverskog i softverskog. Prilikom nabavke same ploče potrebno je proučiti kakve specifikacije treba da ima. Nakon toga, vrijeme se posvećuje softverskom dijelu i razvoju koda za projekat.

Jedna od ključnih stvari zaslužnih za uspjeh Arduina je jednostavnosti korištenja istog, odnosno vrijeme potrebno da se savladaju osnove i napravi prvi uspješan projekat. Postoji velika Arduino zajednica koja je uvijek spremna da pomogne u rješavanju problema. Još je važno naglasiti da je dizajn *open-source* što je kao posljedicu dovelo do razvoja raznih varijantni proizvoda i samim tim doprinijelo razvoju Arduina.

Arduino mikrokontroleri se programiraju u Arduino IDE razvojnom okruženju. Navedeno okruženje je besplatno i ima poprilično jednostavnu instalaciju. Radi približno podjednako dobro na računarima sa različitim operativnim sistemima. Razvojni tim Arduina je radio sa pretpostavkom da ljudima nisu toliko važne specifikacije samog mikrokontrolera koliko im je važno da naprave nešto zanimljivo. Na osnovu toga, napravljene su biblioteke koje su specifične za Arduino. One su jednostavne za shvatiti i primijeniti.

Razvojno okruženje omogućava programiranje ne samo Arduino mikrokontrolera nego i mikrokontrolera drugih proizvođača [26]. Uz instalacije određenih programa moguće je programirati i ESP32 čip na Arduino IDE razvojnom okruženju. Obično su potrebne male izmjene koda u slučaju kompatibilnosti koda i sa Arduino mikrokontrolerom i sa ESP mikrokontrolerom. Izgled Arduino IDE razvojnog okruženja može se vidjeti na slici 3.1, izgled razvojnog okruženja za slučaj programiranja Arduina (lijevo) i za slučaj programiranja ESP-a (desno). Razlika se vidi u donjem desnom dijelu Arduino IDE prozora, gdje su označene specifikacije razvojne ploče i na kojem portu su povezane sa računarom.

Arduino programski jezik je veoma sličan programskom jeziku C. C koristi proceduralnu jezičku sintaksu koju kompajler treba procesirati kako bi preveo sa jezika koji koriste ljudi jezik koji je razumljiv mašinama. Postoje neki dijelovi jezika koji su teško razumljivi i komplikovani. Iz tog razloga Arduino biblioteka posjeduje jednostavan set funkcija koji značajno olakšavaju programiranje (kao što su pinMode(), digitalWrite(), delay()). Navedene biblioteke su ustvari napisane u programskom jeziku C++ i to je jedan od razloga što se Arduino ne smatra u potpunosti C jezikom.



Slika 3.1: Izgled Arduino razvojnog okruženja za Arduino Uno (lijevo) i ESP32-DevKitC (desno)

Program 3.1: Blinkanje LED sa avr-libc

```
#include <avr/io.h>
#include <util/delay.h>

int main(void) {
  while (1) {
    PORTB = 0x20;
    _delay_ms(1000);
    PORTB = 0x00;
    _delay_ms(1000);
  }
  return 1;
}
```

Program 3.2: Blinkanje LED sa Arduinom

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Prethodno navedeni kod je vjerovatno najčešće korišten kod u Arduino programiranju, a to je blinkanje LED. Navedeni su kodovi u slučaju korištenja Arduino biblioteke i bez korištenja iste. Prednost Arduino razvojnog okruženja je u tome što je u potpunosti kompatibilno sa C/C++ kodom napisanog koristeći avr-libc biblioteku, te koristeći GCC *GNU Compiler Collec-*

tion kompajler. Oba koda su kompatibilna sa Arduino razvojnim okruženjem i na isti način se prebacuju na razvojnu ploču.

Razlika između dva navedena primjera je u tome što prvi primjer zahtjeva 210 bajta, a drugi 1010 bajta. Negativna strana Arduino programiranja je u memorijskoj efikasnosti. Označavanje digitalnog pina na koji je povezana LED je lakše korištenjem brojem pina 13, nego korištenje heksadecimalne adrese 0x20 na PORTB. Sa druge strane, ova jednostavnost je prednost Arduino programiranja [26].

Kod programa razvijenog za upotrebu aplikacije sa ESP32 čipom može se naći u prilogu pod imenom *WiFi ESP*, a kod za glavni program koji povezuje sve proučavane komponente se može naći pod imenom *Glavni program ESP*.

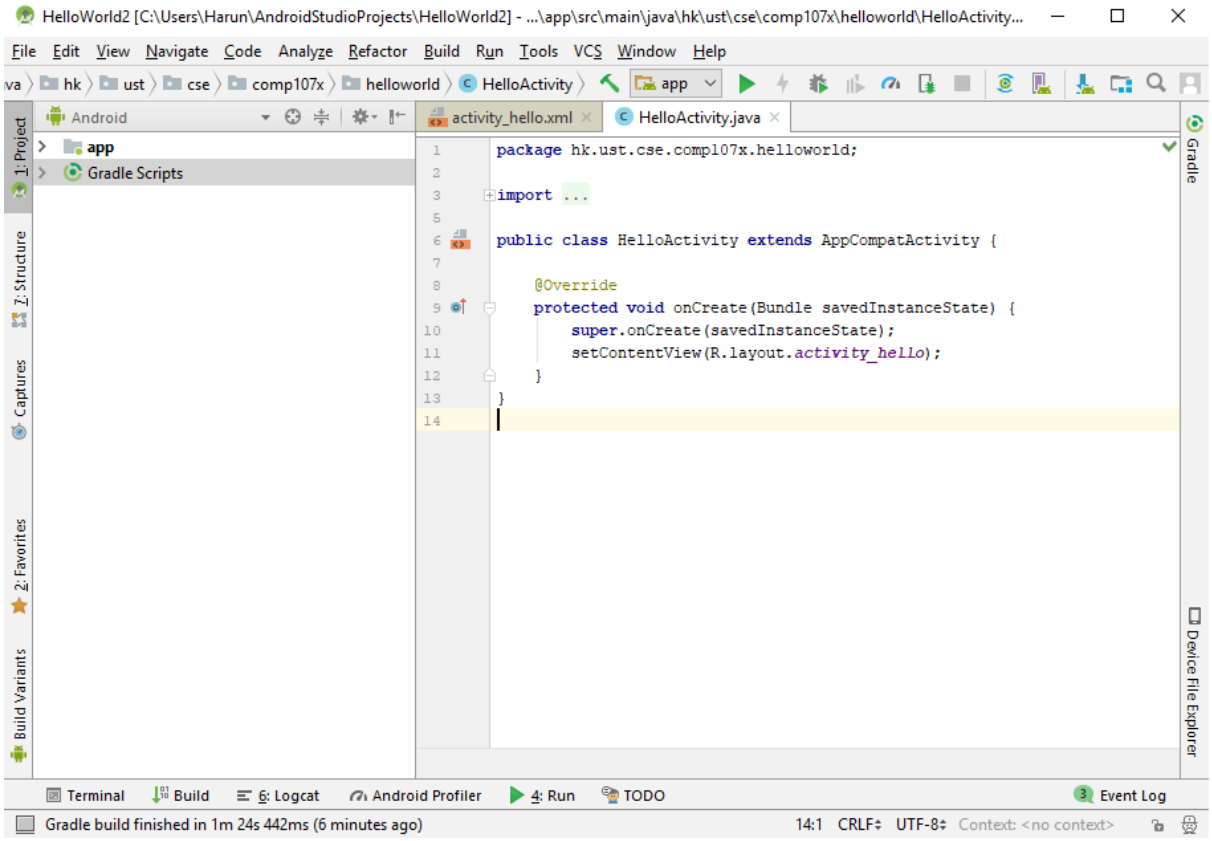
3.2 Android programiranje

Android je *open source* operativni sistem kojeg je napravio Google sa partnerima. Koristi se u preko milijardu uređaja i samim tim je najvažnija platforma za razvijanje aplikacija. Iako su se neke funkcionalnosti pojavile i ranije, Android je prvo okruženje koje je sadržavalo sljedeće:

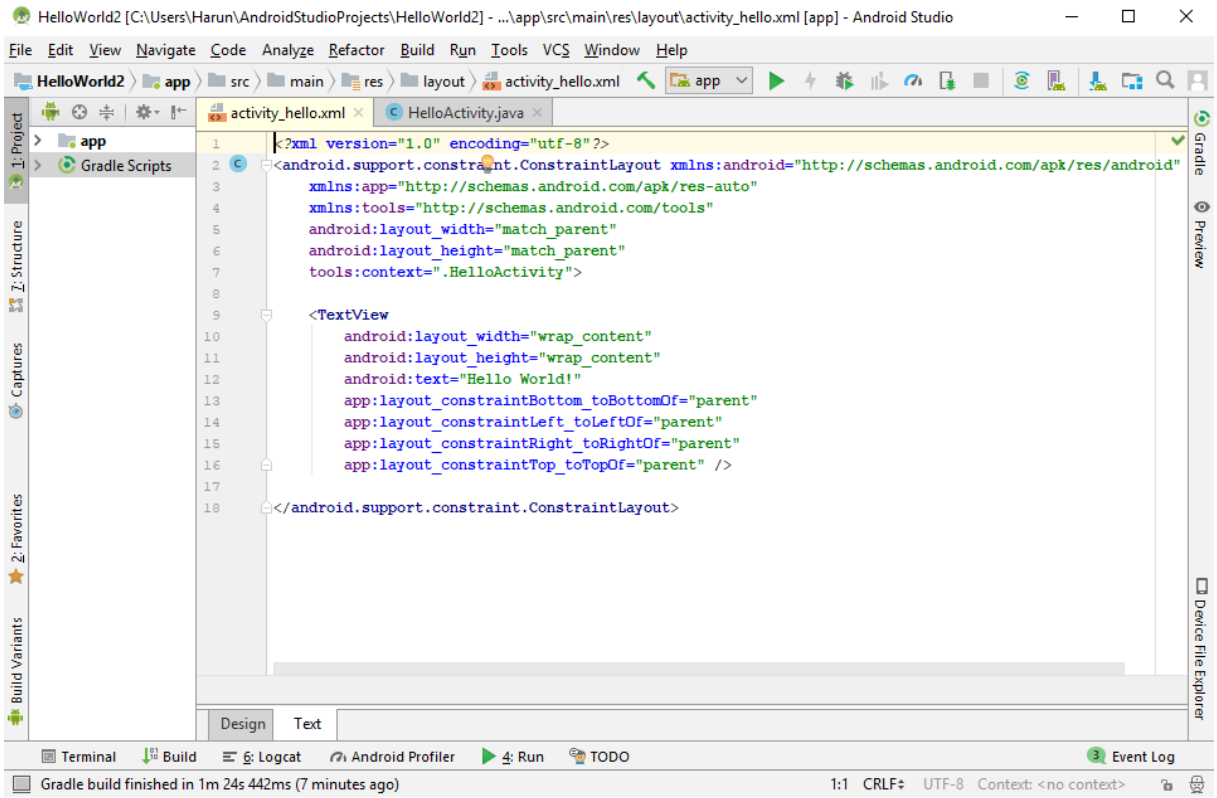
- besplatnu razvojnu platformu baziranu na Linux-u koja je *open source* - omogućilo jeftino korištenje i razvijanje platforme,
- arhitektura bazirana na komponentama - dijelovi jedne aplikacije mogu se koristiti i u svrhe koje nisu prvobitno planirane i moguća promjena ugrađenih komponenti vlastitim verzijama,
- automatsko upravljanje životnog ciklusa aplikacije (eng. *application life cycle*) - programi su međusobno izolirani sa više sigurnosnih slojeva. Krajnji korisnik ne mora brinuti o tome koju aplikaciju treba ugasiti ukoliko hoće da pokrene neku novu. Android je optimiziran za malu potrošnju,
- visok kvalitet slike i zvuka - otvorilo put ka razvoju kvalitetnijih aplikacija,
- prenosivost - razvijeni programi se mogu koristiti na ARM, x86 i drugim arhitekturama. Podržani su razni uređaji za komunikaciju: tastature, miševi, daljinski upravljači i slično.

Do 2013. godine većina developera koristila je Eclipse IDE razvojno okruženje i Android Development Tools. Zatim je Google, u maju iste godine, predstavio novo razvojno okruženje - Android Studio [27]. Pravljenje aplikacije ”Hello World!” je poprilično jednostavno. Kod koji se koristi za navedenu aplikaciju i izgled Android Studio razvojnog okruženja može se vidjeti na slikama 3.2 i 3.3.

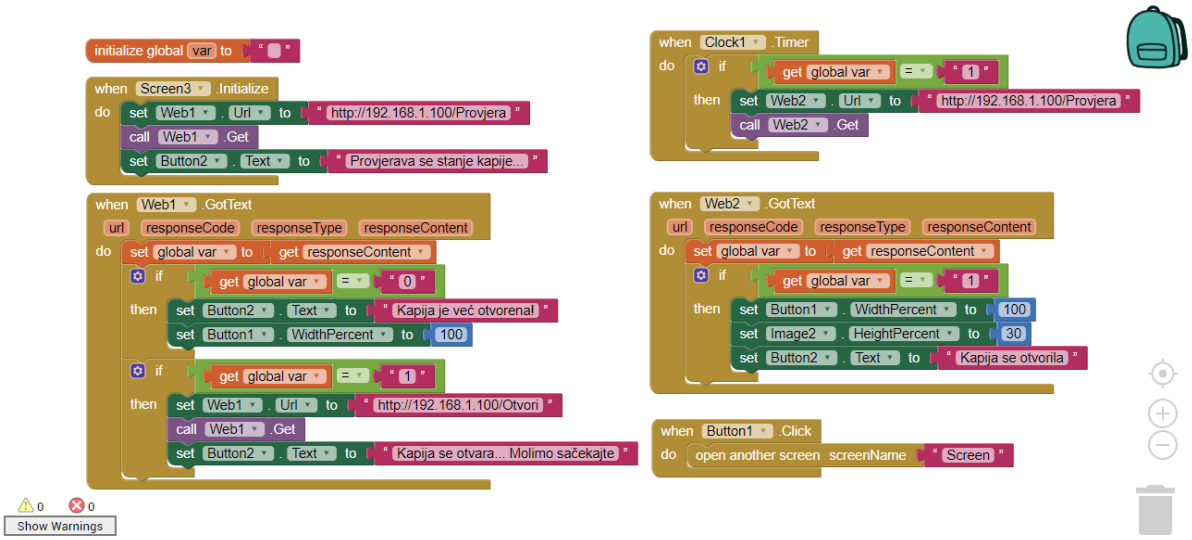
Prilikom pravljenja aplikacije koja se može povezati sa ESP32 čipom korištena je stranica *http://appinventor.mit.edu/explore/* na kojoj se aplikacija kreira pomoću blokova. Primjer blokova formiranih za jedan od ekrana aplikacije može se vidjeti na slici 3.4, a primjer izgleda tri različita ekrana na slici 3.5.



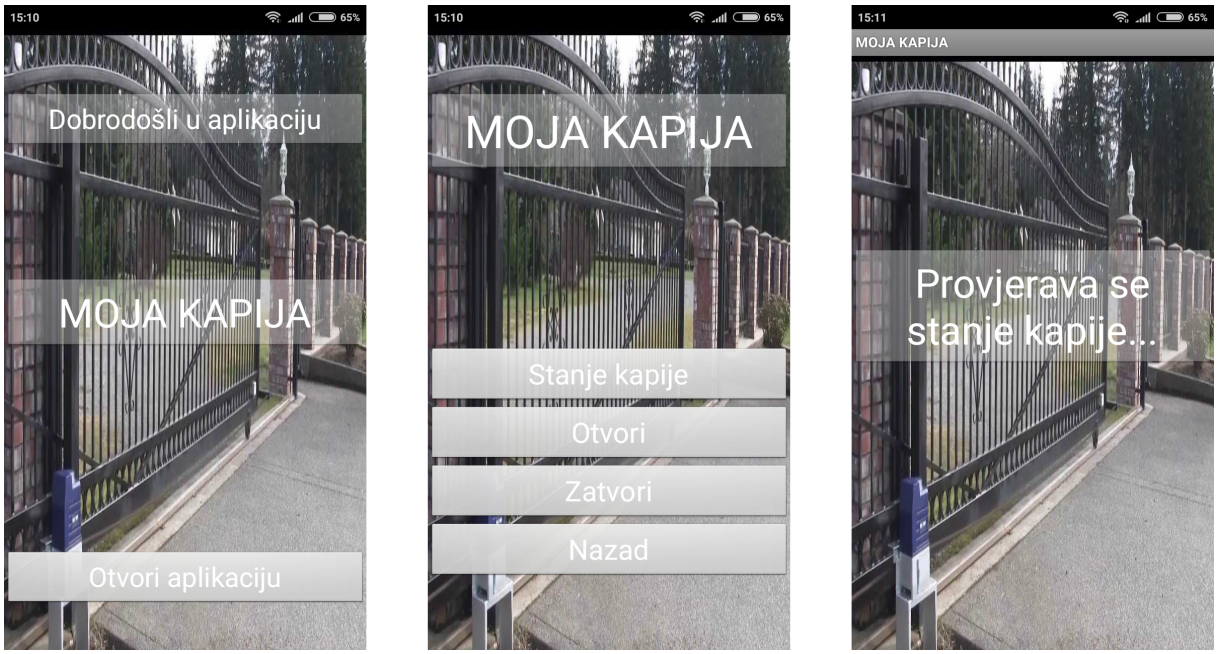
Slika 3.2: Android razvojno okruženje sa kodom za aplikaciju ”Hello World!” - .java



Slika 3.3: Android razvojno okruženje sa kodom za aplikaciju ”Hello World!” - .xml



Slika 3.4: Primjera rasporeda blokova za Android aplikaciju



Slika 3.5: Primjer izgleda ekrana razvijene aplikacije

Poglavlje 4

Eksperimenti i rezultati

Prilikom provjere rada komponenti u prvoj fazi testiranja korištena je Arduino razvojna ploča. Razlog rješavanja problema na ovaj način je činjenica da se mnogo više informacija i podrške može naći za Arduino razvojne ploče nego za ESP32. Nakon što je izvršena provjera i testiranje na Arduinu izvršen je prelazak na ESP32 čip.

4.1 RF prijemnik SRX882 i predajnik STX882

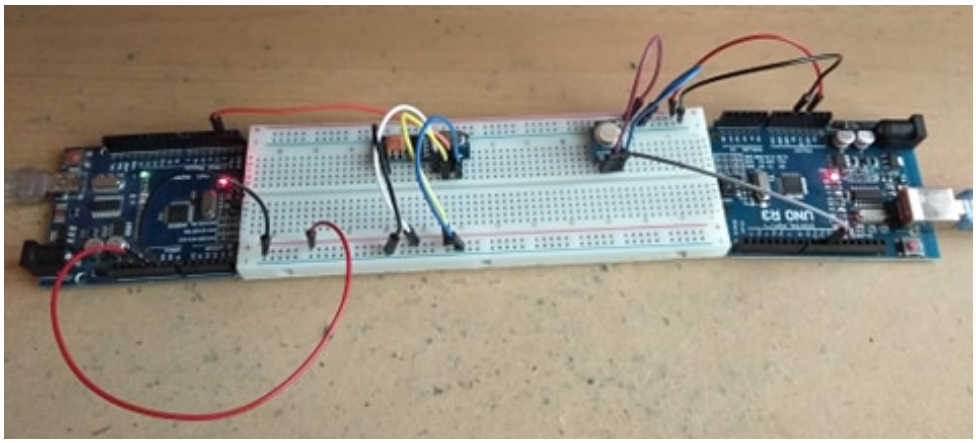
Slika povezivanja RF prijemnika SRX882 i predajnika STX882 može se vidjeti na slici 4.1.

Kodovi korišteni za ovaj slučaj mogu se naći u prilogu pod imenom *RF primopredajnik*. Nakon pokretanja navedenog koda otvori se *Serial Monitor* kako bi se pratio rad sheme i koda. Rezultat uspješne komunikacije se može vidjeti na slici 4.2.

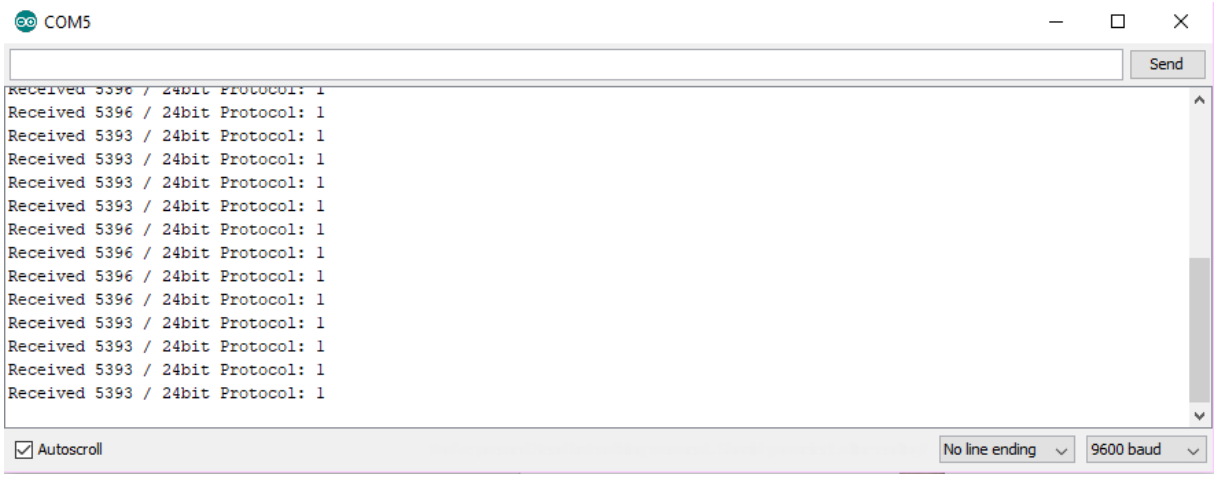
Može se primijetiti da se u navedenom kodu šalju samo dvije različite vrijednosti podataka. U prvom slučaju 5393, a u drugom 5396.

Prilikom navedenog eksperimenta korišten je i uređaj *DVB-T+FM+DAB dongle* koji se može koristiti kao prijemnik signala. Navedeni uređaj se koristio u svrhu proučavanja spektra signala prije korištenja predajnika STX882 i nakon korištenja istog. Za prikaz spektra koristio se programski paket *Touchstone RF Spectrum Analyzer*.

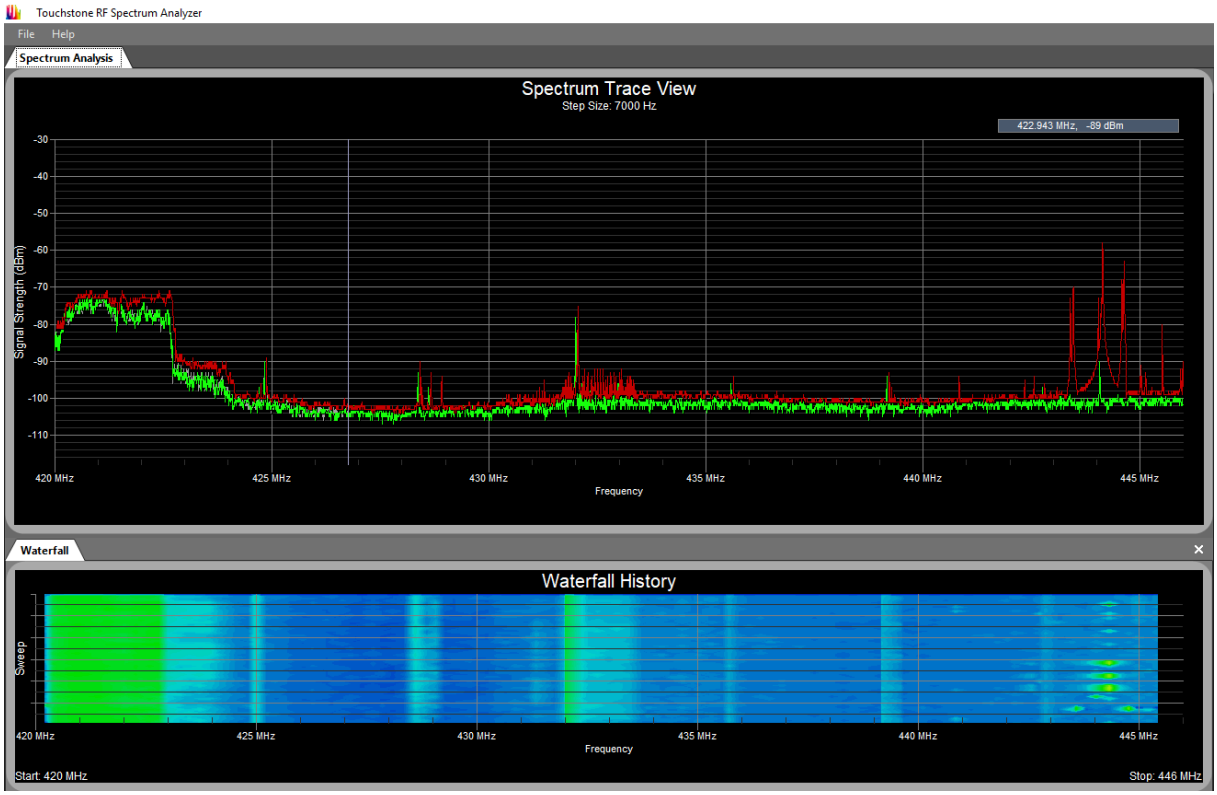
Na slikama 4.3 i 4.4 može se vidjeti razlika u spektru na posmatranoj frekvenciji. Kao što se i očekivalo, pojavila se promjena spektra na frekvenciji od 433.93 MHz. Pored ove najdominantnije promjene pojavila su se još dva značajna *peek*-a na frekvencijama 431.3 MHz



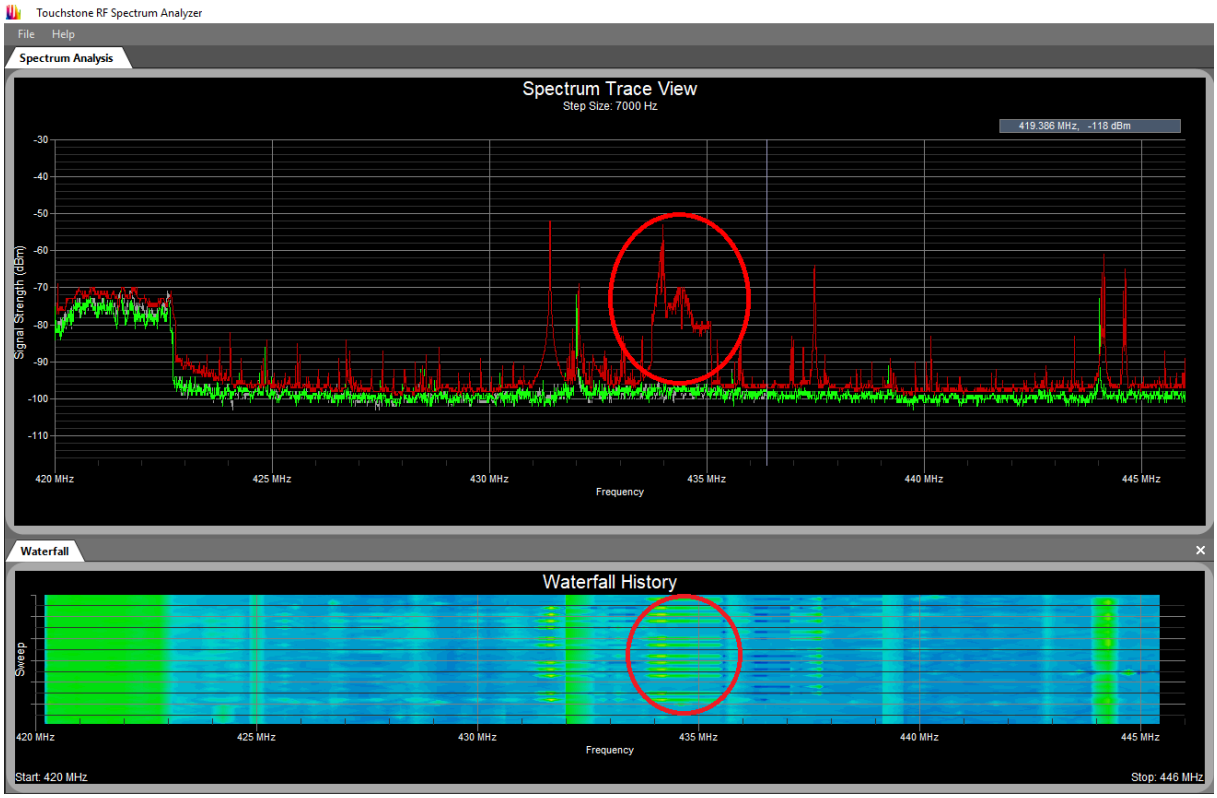
Slika 4.1: Slika povezivanja RF prijemnika SRX882 i predajnika STX882



Slika 4.2: Ispis na *Serial Monitor*-u nakon uspješne komunikacije RF prijemnika i predajnika



Slika 4.3: Spektar prije korištenja predajnika STX882



Slika 4.4: Spektar nakon korištenja predajnika STX882

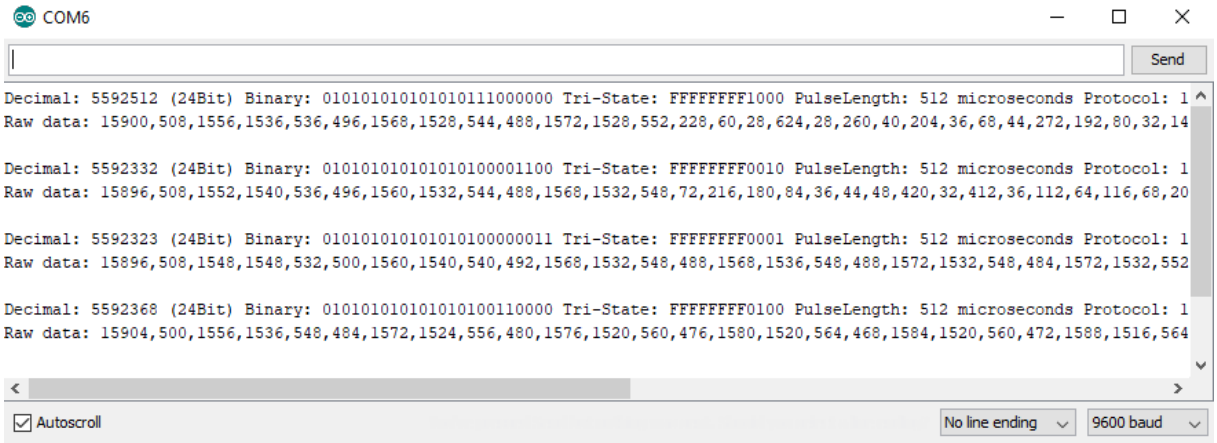
i 437.5 MHz, te jedan manje značajan na frekvenciji od 432 MHz. Također, može se primijetiti distorzija koja je unesena na posmatranom opsegu frekvencija uslijed djelovanja RF predajnika (odnosno pojava velikog broja *peek*-ova na cijelom posmatranom opsegu).

Pored spektra određenog opsega frekvencija, na posmatranim slikama može se uočiti i *Waterfall History*. *Waterfall History* na y-osi pokazuje vrijeme, dok se na x-osi nalaze frekvencije. Pomoću *Waterfall History* može se posmatrati pojavljivanje signala na određenoj frekvenciji u zavisnosti od vremena. Signal koji je prisutan na obje slike na frekvencijama od 420 MHz do 425 MHz ima zelenu boju na *Waterfall History* prikazu zbog konstantnog prisustva istog. Na frekvencijama gdje nema signala, boja na *Waterfall History* prikazu je plava. Za slučaj kada se povremeno pojavljuje signal (predajnik SRX882 šalje signale), može se primijetiti pojavljivanje zelene boje u pojedinim trenucima u vremenu.

4.2 RF prijemnik SRX882 i daljinski upravljač

Nakon uspješne provjere rada prijemnika SRX882, posmatrat će se rad daljinskog upravljača u cilju realizacije daljinskog upravljanja kapijom. Korišteni daljinski ima 4 različita dugmeta i pritiskom bilo kojeg od njih dobija se drugačija vrijednost na prijemnoj strani. Kod korišten za provjeru rada daljinskog upravljača može se naći u prilogu pod imenom *RF prijemnik i daljinski*.

Nakon pokretanja koda i izvršene provjere ispravnosti komponenti, dobiju se vrijednosti u *Serial Monitor*-u kao na slici 4.5. Mogu se primijetiti četiri različita slučaja primljenih vrijednosti koje odgovaraju dugmadima sa daljinskog upravljača. Prva vrijednost koja se ispisuje je decimalna vrijednost dodijeljena poslanom signalu na predajnoj strani. Nakon toga se ispisuje binarna reprezentacija istog broja u 24-bitnom obliku. Sljedeći ispis predstavlja *tri-state* vrijed-



Slika 4.5: Ispis na *Serial Monitor*-u nakon uspješne komunikacije RF prijemnika i daljinskog upravljača

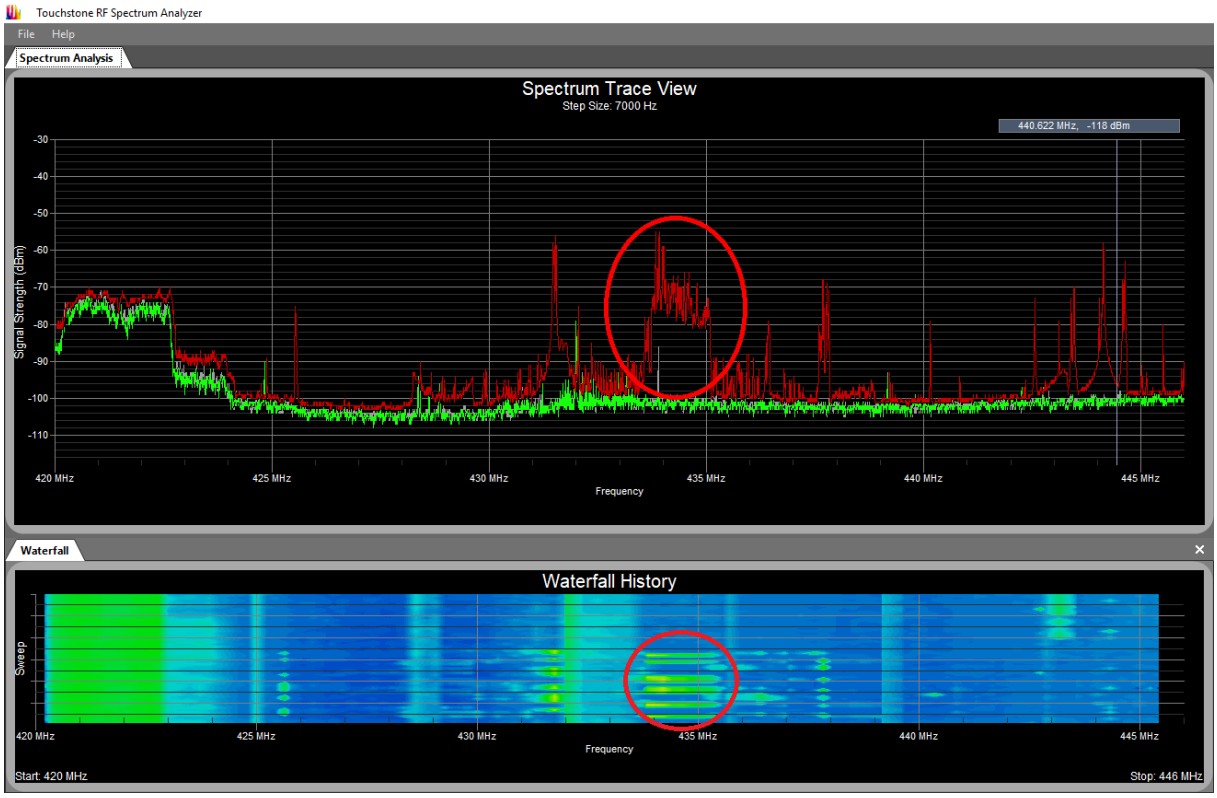
nost. To je vrijednost koja se predstavlja sa tri različita simbola (u ovom slučaju sa 0, 1 i F). Zatim se prikazuje trajanje impulsa u mikrosekundama. Na kraju reda se ispisuje protokol koji se koristi (o protokolima koji se mogu koristiti bilo je više govora u poglavlju). U drugom redu se prikazuju sirovi podaci (eng. *Raw data*). Sirovi podaci su podaci koji nisu obrađeni, odnosno predstavljaju podatke u onom obliku u kojem se šalju kroz medij.

U toku navedene provjere iz prethodnog koraka vršeno je posmatranje spektra. Dobijene su slične vrijednosti spektra kao i na slici 4.4. Razlika se može primijetiti posmatranjem *Waterfall History* prikaza. Vidi se mnogo rjeđe pojavljivanje zelenih dijelova u pojedinim trenucima vremena na frekvenciji od 433.93 MHz zbog značajno sporijeg slanja podataka. Činjenica je da je ljudima potrebno mnogo više vremena za slanje podataka pritiskanjem dugmeta nego što je potrebno RF predajniku da pošalje jednu seriju podataka (koja se može poslati u djeliću sekunde).

4.3 Povezivanje RF prijemnika SRX882, daljinskog upravljača, LCD ekrana i L298N drajvera

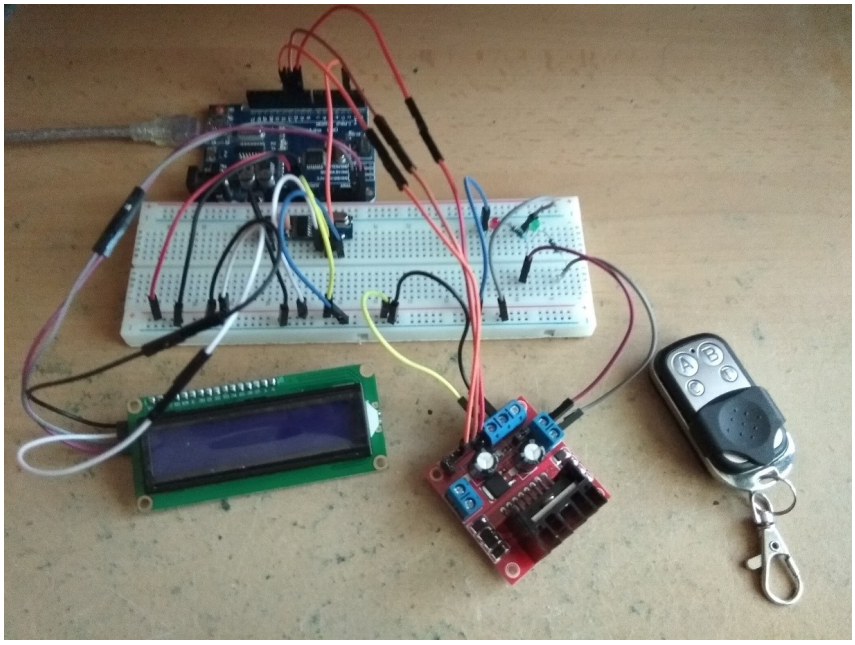
U cilju smanjenja broja potrebnih pinova za spajanje LCD ekrana sa mikrokontrolerom, korišten je *I2C/SPI LCD backpack*. Pomoću navedene komponente smanjuje se broj potrebnih pinova sa osam na dva. *LCD backpack* se nalazi sa druge strane LCD ekrana.

Ovaj eksperiment predstavlja sljedeći korak u razvoju uređaja za elektroničko upravljanje kapijom. Nakon što se uspješno realizovao rad LCD ekrana sa SRX882 prijemnikom i daljinskim upravljačem dodaje se i L298N drajver za motor. Iz razloga što se može javiti potreba za većim strujama prilikom rada DC motora na izlazu iz L298N, korištene su LED za signalizaciju izlaza iz L298N (odnosno provjere rada drajvera). Pritiskom dugmeta A kapija se otvara, odnosno dolazi do aktivacije zelene LED. Pritiskom dugmeta B kapija se zatvara te se aktivira crvena LED. Primjer ispisa na LCD ekran može se vidjeti na slici 4.9. Kod korišten za ovaj slučaj se može naći pod imenom *RF, LCD i L298N*.

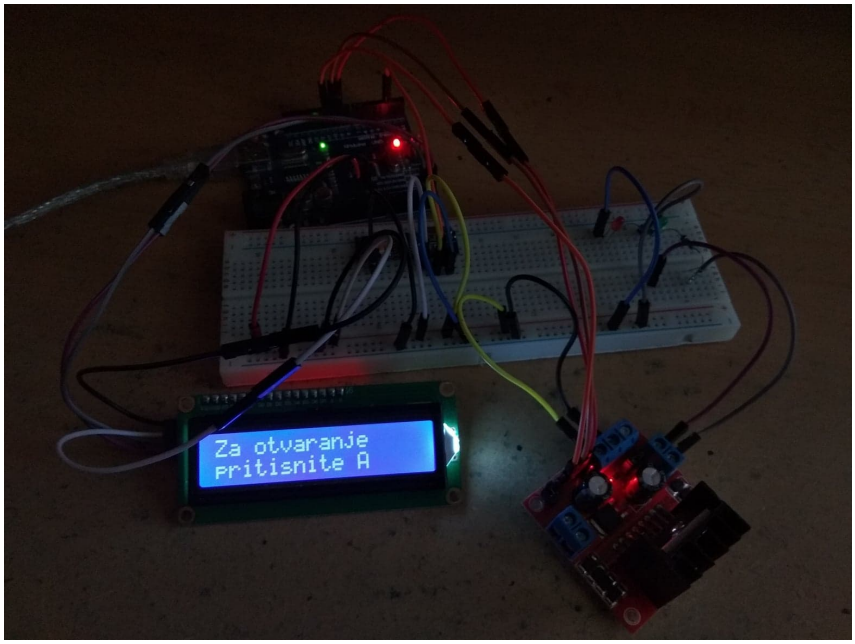


Slika 4.6: Spektar nakon korištenja daljinskog upravljača

Slika 4.7: Fizički izgled *I2C/SPI LCD backpack* (slika preuzeta iz [11]



Slika 4.8: Slika spajanja navedenog eksperimenta



Slika 4.9: Ispis na LCD ekranu

Zaključak

U ovom radu prezentiran je dizajn elektroničke strukture za daljinsko upravljanje kapijom. Predstavljeno rješenje posjeduje mogućnost upravljanja kapijom pomoću tastature, daljinskog upravljača i mobilne aplikacije. Više načina komunikacije omogućava pouzdanije upravljanje u slučaju da jedan od načina komunikacije ne bude funkcionalan. Realizovano rješenje ne zahtijeva velike ispravke, ni hardverske ni softverske, u slučaju dodavanja novih funkcionalnosti.

Smjernice za budući rad

Korišteni čip ESP32 posjeduje ugrađenu Bluetooth komunikaciju koja bi se mogla iskoristiti kao novi način komunikacije u slučaju loših rezultata sa nekim od realizovanih načina komuniciranja.

Elektronička struktura posjeduje odgovarajuće senzore koji se koriste za prevenciju nesreće, kako bi se prekinuo rad kapije u slučaju da postoji određena prepreka prilikom otvaranja ili zatvaranja kapije. Međutim, postoje senzori čija bi upotreba davala bolje rezultate u primjenama kao što je pokretna kapija. Mogli bi se implementirati optički senzori ili senzori za detekciju pokreta.

Moguće je ostvariti određena poboljšanja ukoliko bi se koristio čip koji ima bolje karakteristike, kao što su temperaturna izdržljivost, veća memorija, veći broj pinova i slično.

Prilozi

Prilog A

Programski kodovi

A.1 Kodovi korišteni za eksperimente

A.1.1 RF primopredajnik

Kodovi za RF prijemnik SRX882 i predajnik ST882 su preuzeti kao već postojeći primjeri iz biblioteke *rc-switch*. Kod za RF prijemnik se može naći pod imenom *ReceiveDemo Advanced*, a za RF predajnik *SendDemo*. Izvršene su male izmjene navedenih kodova.

Program A.1: RF SRX882 prijemnik - glavni program

```
#include <RCSwitch.h>

RCSwitch mySwitch = RCSwitch();

void setup() {
  Serial.begin(115200);
  mySwitch.enableReceive(27);
}

void loop() {
  if (mySwitch.available()) {
    output(mySwitch.getReceivedValue(), mySwitch.
        getReceivedBitlength(), mySwitch.getReceivedDelay(),
        mySwitch.getReceivedRawdata(),mySwitch.getReceivedProtocol()
        );
    mySwitch.resetAvailable();
  }
}
```

Program A.2: RF SRX882 prijemnik - funkcija

```
static const char* bin2tristate(const char* bin);
static char * dec2binWzerofill(unsigned long Dec, unsigned int
    bitLength);

void output(unsigned long decimal, unsigned int length, unsigned int
     delay, unsigned int* raw, unsigned int protocol) {

  if (decimal == 0) {
    Serial.print("Unknown␣encoding.");
  } else {
    const char* b = dec2binWzerofill(decimal, length);
```

```
10      Serial.print("Decimal:␣");
11      Serial.print(decimal);
12      Serial.print("␣(");
13      Serial.print( length );
14      Serial.print("Bit)␣Binary:␣");
15      Serial.print( b );
16      Serial.print("␣Tri-State:␣");
17      Serial.print( bin2tristate( b) );
18      Serial.print("␣PulseLength:␣");
19      Serial.print(delay);
20      Serial.print("␣microseconds");
21      Serial.print("␣Protocol:␣");
22      Serial.println(protocol);
23    }
24
25    Serial.print("Raw␣data:␣");
26    for (unsigned int i=0; i<= length*2; i++) {
27      Serial.print(raw[i]);
28      Serial.print(",");
29    }
30    Serial.println();
31    Serial.println();
32  }
33
34  static const char* bin2tristate(const char* bin) {
35    static char returnValue[50];
36    int pos = 0;
37    int pos2 = 0;
38    while (bin[pos]!='\0' && bin[pos+1]!='\0') {
39      if (bin[pos]=='0' && bin[pos+1]=='0') {
40        returnValue[pos2] = '0';
41      } else if (bin[pos]=='1' && bin[pos+1]=='1') {
42        returnValue[pos2] = '1';
43      } else if (bin[pos]=='0' && bin[pos+1]=='1') {
44        returnValue[pos2] = 'F';
45      } else {
46        return "not␣applicable";
47      }
48      pos = pos+2;
49      pos2++;
50    }
51    returnValue[pos2] = '\0';
52    return returnValue;
53  }
54
55  static char * dec2binWzerofill(unsigned long Dec, unsigned int
      bitLength) {
56    static char bin[64];
57    unsigned int i=0;
58
59    while (Dec > 0) {
60      bin[32+i++] = ((Dec & 1) > 0) ? '1' : '0';
61      Dec = Dec >> 1;
62    }
63
64    for (unsigned int j = 0; j< bitLength; j++) {
65      if (j >= bitLength - i) {
66        bin[j] = bin[ 31 + i - (j - (bitLength - i)) ];
```

```
67      } else {
68        bin[j] = '0';
69      }
70    }
71    bin[bitLength] = '\0';
72
73    return bin;
74  }
```

Program A.3: RF STX882 predajnik

```
1   #include <RCSwitch.h>
2
3   RCSwitch mySwitch = RCSwitch();
4
5   void setup() {
6
7     Serial.begin(9600);
8
9     // Predajnik se nalazi na pinu 10
10    mySwitch.enableTransmit(10);
11
12    // Promjena trajanja impulsa
13    // mySwitch.setPulseLength(320);
14
15    // Izbor protokola
16    // mySwitch.setProtocol(2);
17
18    // Broj ponavljanja transmisije podataka
19    // mySwitch.setRepeatTransmit(15);
20
21  }
22
23  void loop() {
24
25    mySwitch.switchOn("11111", "00010");
26    delay(1000);
27    mySwitch.switchOff("11111", "00010");
28    delay(1000);
29
30    /* Sada se biraju decimalne vrijednosti */
31    mySwitch.send(5393, 24);
32    delay(1000);
33    mySwitch.send(5396, 24);
34    delay(1000);
35
36    /* Definisanje binarnih vrijednosti */
37    mySwitch.send("000000000001010100010001");
38    delay(1000);
39    mySwitch.send("000000000001010100010100");
40    delay(1000);
41
42    /* Definisanje tri-state koda*/
43    mySwitch.sendTriState("00000FFF0F0F");
44    delay(1000);
45    mySwitch.sendTriState("00000FFF0FF0");
46    delay(1000);
47
```

```
  delay(20000);
}
```

A.1.2 RF prijemnik i daljinski

Program A.4: RF prijemnik i daljinski

```
#include <RCSwitch.h>

RCSwitch mySwitch = RCSwitch();

void setup() {
  Serial.begin(9600);
  mySwitch.enableReceive(0);  // oznaka pina brojem 0 implicira da
      se korist pin broj 2 na Arduinu
}

void loop() {

  if (mySwitch.available()) {
    int received = mySwitch.getReceivedValue();

    if (received == 0) {
      Serial.print("Unknown encoding");
    } else {
      Serial.print("Received ");
      received = mySwitch.getReceivedValue();
      Serial.println(received);
    }

    if (received == 21952) Serial.println("Pritisnuto je dugme A");
    else if (received == 21772) Serial.println("Pritisnuto je dugme
        B");
    else if (received == 21763) Serial.println("Pritisnuto je dugme
        C");
    else Serial.println("Pritisnuto je dugme D");

    mySwitch.resetAvailable();
  }
}
```

A.1.3 RF, LCD i L298N

Program A.5: RF LCD i L298N

```
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#include <RCSwitch.h>

LiquidCrystal_I2C  lcd(0x3f,2,1,0,4,5,6,7);
RCSwitch mySwitch = RCSwitch();

const uint8_t lcd_ekran = 3;
const uint8_t enable_DC = 9;
```

```
11 const uint8_t forward = 10;
12 const uint8_t reverse = 11;
13
14 void setup()
15 {
16     lcd.setBacklightPin(lcd_ekran,POSITIVE);
17     lcd.begin(16, 2);
18     lcd.clear();
19
20     mySwitch.enableReceive(0);
21
22     pinMode(enable_DC, OUTPUT);
23     pinMode(forward, OUTPUT);
24     pinMode(reverse, OUTPUT);
25
26     Serial.begin(9600);
27 }
28
29 int br = 0;
30
31 void loop() {
32
33   if (br == 0) {
34     lcd.setCursor(0,0);
35     lcd.print("Za␣otvaranje");
36     delay(1000);
37     lcd.setCursor(0,1);
38     lcd.print("pritisnite␣A");
39     delay(1000);
40     lcd.clear();
41     lcd.setCursor(0,0);
42     lcd.print("Za␣zatvaranje");
43     delay(1000);
44     lcd.setCursor(0,1);
45     lcd.print("pritisnite␣B");
46     delay(1000);
47     lcd.clear();
48   }
49   delay(50);
50   br++;
51   Serial.println(br);
52   if (br == 500) br = 0;
53
54   if (mySwitch.available()) {
55
56     int received = mySwitch.getReceivedValue();
57
58     if (received == 0) {
59       Serial.print("Nepoznato!");
60     } else {
61       Serial.print("Primljeno:␣");
62       received = mySwitch.getReceivedValue();
63       Serial.println(received);
64     }
65
66     if (received == 21952) {
67       lcd.clear();
68       lcd.setCursor(0,0);
```

```
      lcd.print("Pritisnuli␣ste");
      delay(500);
      lcd.setCursor(0,1);
      lcd.print("dugme␣A");
      delay(500);
      lcd.clear();
      Serial.println("Pritisnuto␣je␣dugme␣A");
      digitalWrite(forward, HIGH);
      digitalWrite(reverse, LOW);
      analogWrite(enable_DC, 255);
      delay(3000);
      digitalWrite(forward, LOW);
    }
    else if (received == 21772) {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Pritisnuli␣ste");
      delay(500);
      lcd.setCursor(0,1);
      lcd.print("dugme␣B");
      delay(500);
      lcd.clear();
      delay(500);
      Serial.println("Pritisnuto␣je␣dugme␣B");
      digitalWrite(forward, LOW);
      digitalWrite(reverse, HIGH);
      analogWrite(enable_DC, 255);
      delay(3000);
      digitalWrite(reverse, LOW);
    }
    else if (received == 21763) {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Pritisnuli␣ste");
      delay(500);
      lcd.setCursor(0,1);
      lcd.print("dugme␣C");
      delay(500);
      lcd.clear();
      delay(500);
      Serial.println("Pritisnuto␣je␣dugme␣C");
    }
    else {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Pritisnuli␣ste");
      delay(500);
      lcd.setCursor(0,1);
      lcd.print("dugme␣D");
      delay(500);
      lcd.clear();
      delay(500);
      Serial.println("Pritisnuto␣je␣dugme␣D");
    }

    mySwitch.resetAvailable();
  }
}
```

A.1.4 WiFi ESP

Program A.6: WiFi ESP

```
#include <WiFi.h>

const uint8_t enable_DC = 4;
const uint8_t forward = 2;
const uint8_t reverse = 15;
const uint8_t sensor_close = 18;
const uint8_t sensor_open = 34;

String  ClientRequest;
IPAddress staticIP390_3(192,168,1,100);
IPAddress gateway390_3(192,168,1,1);
IPAddress subnet390_3(255,255,255,0);

WiFiServer server(80);
WiFiClient client;
String myresultat;

String ReadIncomingRequest() {
while(client.available()) {
  ClientRequest = (client.readStringUntil('\r'));
     if ((ClientRequest.indexOf("HTTP/1.1")>0)&&(ClientRequest.
        indexOf("/favicon.ico")<0)) {
      myresultat = ClientRequest;
    }
  }
return myresultat;
}

void setup()  {

  ClientRequest = "";
  Serial.begin(115200);

  pinMode(enable_DC, OUTPUT);
  pinMode(forward, OUTPUT);
  pinMode(reverse, OUTPUT);
  WiFi.disconnect();
  delay(3000);
  Serial.println("START");
  WiFi.begin("MojaTV_Net_216083","11335599");
  while ((!(WiFi.status() == WL_CONNECTED))){
    delay(300);
    Serial.print("..");
  }

  Serial.println("Connected");
  WiFi.config(staticIP390_3, gateway390_3, subnet390_3);
  Serial.println("Your␣IP␣is");
  Serial.println((WiFi.localIP()));
  server.begin();

}

int sen_val_close = 0, sen_val_open = 0;
```

```
54 int br=0, closed_app = 0;
55
56 void loop() {
57
58     sen_val_close = digitalRead(sensor_close);
59     sen_val_open = digitalRead(sensor_open);
60     client = server.available();
61
62     if (!client) { return; }
63     while(!client.available()){  delay(1); }
64     ClientRequest = (ReadIncomingRequest());
65     ClientRequest.remove(0, 5);
66     ClientRequest.remove(ClientRequest.length()-9,9);
67
68     if (ClientRequest == "Otvori") {
69       digitalWrite(forward, HIGH);
70       digitalWrite(reverse, LOW);
71       digitalWrite(enable_DC, HIGH);
72       while (sen_val_open) {
73         sen_val_open = digitalRead(sensor_open);
74       }
75       if (sen_val_open == 0) {
76         digitalWrite(forward, LOW);
77         closed_app = 0;
78       }
79       client.println("HTTP/1.1␣200␣OK");
80       client.println("Content-Type:␣text/html");
81       client.println("");
82       client.stop();
83       delay(1);
84     }
85
86     if (ClientRequest == "Zatvori") {
87       digitalWrite(forward, LOW);
88       digitalWrite(reverse, HIGH);
89       digitalWrite(enable_DC, HIGH);
90       while (sen_val_close) {
91         sen_val_close = digitalRead(sensor_close);
92       }
93       if (sen_val_close == 0) {
94         digitalWrite(reverse, LOW);
95         closed_app = 1;
96       }
97       client.println("HTTP/1.1␣200␣OK");
98       client.println("Content-Type:␣text/html");
99       client.println("");
100      client.stop();
101      delay(1);
102    }
103
104    if (ClientRequest == "Provjera") {
105      client.println("HTTP/1.1␣200␣OK");
106      client.println("Content-Type:␣text/html");
107      client.println("");
108      client.println(closed_app);
109      client.stop();
110      delay(1);
111    }
```

```

    client.flush();
}
```

A.1.5 Glavni program ESP

Program A.7: Glavni program ESP

```
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>
#include <RCSwitch.h>
#include <Wire.h>

const uint8_t enable_DC = 4;
const uint8_t forward = 2;
const uint8_t reverse = 15;
const uint8_t sensor_close = 18;
const uint8_t sensor = 19;
const uint8_t sensor_open = 34;

const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {13, 12, 14, 26};
byte colPins[COLS] = {25, 33, 32, 3};

RCSwitch mySwitch = RCSwitch();
LiquidCrystal_I2C lcd(0x3F, 16, 2);
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins
    , ROWS, COLS);

void setup() {

  Serial.begin(115200);

  lcd.init();
  lcd.init();
  lcd.setBacklight(HIGH);

  mySwitch.enableReceive(27);

  pinMode(enable_DC, OUTPUT);
  pinMode(forward, OUTPUT);
  pinMode(reverse, OUTPUT);

  pinMode(sensor_close, INPUT);
  pinMode(sensor, INPUT);
  pinMode(sensor_open, INPUT);

}

char sifra[4] = {'0', '0', '0', '0'};
```

```
49  char nova_sifra[4] = {'0', '0', '0', '0'};
50  int br = 0, i = 0;
51  int sen_val_close = 0, sen_val = 0, sen_val_open = 0;
52  bool closed = true;
53
54  void loop() {
55
56    char uneseni_znak = customKeypad.getKey();
57
58      if (br == 0) {
59        lcd.setCursor(0,0);
60        lcd.print("DALJINSKI␣");
61        lcd.setCursor(0,1);
62        lcd.print("UPRAVLJAC␣");
63        delay(1000);
64        lcd.clear();
65        lcd.setCursor(0,0);
66        lcd.print("Za␣otvaranje");
67        delay(500);
68        lcd.setCursor(0,1);
69        lcd.print("pritisni␣A");
70        delay(1000);
71        lcd.clear();
72        lcd.setCursor(0,0);
73        lcd.print("Za␣zatvaranje");
74        delay(500);
75        lcd.setCursor(0,1);
76        lcd.print("pritisni␣B");
77        delay(1000);
78        lcd.clear();
79        lcd.setCursor(0,0);
80        delay(500);
81
82        lcd.print("TASTATURA");
83        delay(1000);
84        lcd.clear();
85        lcd.print("Za␣unos␣sifre");
86        delay(500);
87        lcd.setCursor(0,1);
88        lcd.print("pritisni␣#");
89        delay(1000);
90        lcd.clear();
91        lcd.setCursor(0,0);
92        lcd.print("Za␣novu␣sifru");
93        delay(500);
94        lcd.setCursor(0,1);
95        lcd.print("pritisni␣*");
96        delay(1000);
97        lcd.clear();
98      }
99      delay(50);
100     br++;
101     if (br == 500) br = 0;
102
103     sen_val_close = digitalRead(sensor_close);
104     sen_val = digitalRead(sensor);
105     sen_val_open = digitalRead(sensor_open);
106
```

```
107     if (!closed && br % 20 == 0) {
108       lcd.setCursor(0,0);
109       lcd.print("Provjerite␣");
110       delay(500);
111       lcd.setCursor(0,1);
112       lcd.print("stanje␣kapije");
113       delay(1000);
114       lcd.clear();
115     }
116
117     if (mySwitch.available()) {
118       int received = mySwitch.getReceivedValue();
119
120       if (received == 0) {
121         Serial.print("Nepoznato!");
122       } else {
123         Serial.print("Primljeno:␣");
124         received = mySwitch.getReceivedValue();
125         Serial.println(received);
126       }
127
128       if (received == 5592512) {
129         lcd.clear();
130         lcd.setCursor(0,0);
131         lcd.print("Pritisnuto␣je");
132         delay(500);
133         lcd.setCursor(0,1);
134         lcd.print("dugme␣A");
135         delay(500);
136         lcd.clear();
137         Serial.println("Pritisnuto␣je␣dugme␣A");
138         lcd.setCursor(0,0);
139         lcd.print("Kapija␣se␣");
140         lcd.setCursor(0,1);
141         lcd.print("otvara");
142         digitalWrite(forward, HIGH);
143         digitalWrite(reverse, LOW);
144         digitalWrite(enable_DC, HIGH);
145         while (sen_val_open && sen_val) {
146           sen_val = digitalRead(sensor);
147           sen_val_open = digitalRead(sensor_open);
148         }
149         if (sen_val_open == 0) {
150           lcd.clear();
151           digitalWrite(forward, LOW);
152           lcd.setCursor(0,0);
153           lcd.print("Kapija␣se");
154           delay(500);
155           lcd.setCursor(0,1);
156           lcd.print("otvorila");
157           delay(1000);
158           lcd.clear();
159           closed = true;
160         }
161         if (sen_val == 0) {
162           lcd.clear();
163           digitalWrite(forward, LOW);
164           lcd.setCursor(0,0);
```

```
165          lcd.print("Prekinuto␣je␣");
166          delay(500);
167          lcd.setCursor(0,1);
168          lcd.print("otvaranje!");
169          delay(1000);
170          lcd.clear();
171          closed = false;
172        }
173      }
174      else if (received == 5592332) {
175        lcd.clear();
176        lcd.setCursor(0,0);
177        lcd.print("Pritisnuto␣je");
178        delay(500);
179        lcd.setCursor(0,1);
180        lcd.print("dugme␣B");
181        delay(500);
182        lcd.clear();
183        lcd.setCursor(0,0);
184        lcd.print("Kapija␣se");
185        lcd.setCursor(0,1);
186        lcd.print("zatvara");
187        Serial.println("Pritisnuto␣je␣dugme␣B");
188        digitalWrite(forward, LOW);
189        digitalWrite(reverse, HIGH);
190        digitalWrite(enable_DC, HIGH);
191        while (sen_val_close && sen_val) {
192          sen_val_close = digitalRead(sensor_close);
193          sen_val = digitalRead(sensor);
194        }
195        if (sen_val_close == 0) {
196          lcd.clear();
197          digitalWrite(reverse, LOW);
198          lcd.setCursor(0,0);
199          lcd.print("Kapija␣se");
200          delay(500);
201          lcd.setCursor(0,1);
202          lcd.print("zatvorila");
203          delay(1000);
204          lcd.clear();
205          closed = true;
206        }
207        if (sen_val == 0) {
208          lcd.clear();
209          digitalWrite(reverse, LOW);
210          lcd.setCursor(0,0);
211          lcd.print("Prekinuto␣je␣");
212          delay(500);
213          lcd.setCursor(0,1);
214          lcd.print("zatvaranje␣kapije!");
215          delay(1000);
216          lcd.clear();
217          closed = false;
218        }
219      }
220      else if (received == 5592323) {
221        lcd.clear();
222        lcd.setCursor(0,0);
```

```
223         lcd.print("Pritisnuto␣je");
224         delay(500);
225         lcd.setCursor(0,1);
226         lcd.print("dugme␣C");
227         delay(500);
228         lcd.clear();
229         delay(500);
230         Serial.println("Pritisnuto␣je␣dugme␣C");
231       }
232       else {
233         lcd.clear();
234         lcd.setCursor(0,0);
235         lcd.print("Pritisnuto␣je");
236         delay(500);
237         lcd.setCursor(0,1);
238         lcd.print("dugme␣D");
239         delay(500);
240         lcd.clear();
241         delay(500);
242         Serial.println("Pritisnuto␣je␣dugme␣D");
243       }
244
245       mySwitch.resetAvailable();
246     }
247
248     if (uneseni_znak == '#') {
249       lcd.setCursor(0,0);
250       lcd.print("Unesi␣sifru:");
251       delay(1000);
252       while(true) {
253         uneseni_znak = NO_KEY;
254         while (uneseni_znak == NO_KEY) uneseni_znak = customKeypad.
              getKey();
255         Serial.println(uneseni_znak);
256         nova_sifra[i] = uneseni_znak;
257         lcd.setCursor(0,1);
258         lcd.print(uneseni_znak);
259         i++;
260         if (i == 4) break;
261       }
262
263       i = 0;
264       lcd.clear();
265       lcd.setCursor(0,0);
266       lcd.print("Unesena␣je");
267       lcd.setCursor(0,1);
268       lcd.print("sifra:␣");
269       lcd.print(nova_sifra[0]);
270       lcd.print(nova_sifra[1]);
271       lcd.print(nova_sifra[2]);
272       lcd.print(nova_sifra[3]);
273       delay(2000);
274       lcd.clear();
275
276       if (nova_sifra[0] == sifra[0] && nova_sifra[1] == sifra[1] &&
            nova_sifra[2] == sifra[2] && nova_sifra[3] == sifra[3]) {
277         lcd.setCursor(0,0);
278         lcd.print("Unesena␣je␣");
```

```
279         lcd.setCursor(0,1);
280         lcd.print("ispravna␣sifra");
281         delay(1000);
282         lcd.clear();
283 
284         lcd.setCursor(0,0);
285         lcd.print("Za␣otvaranje");
286         delay(500);
287         lcd.setCursor(0,1);
288         lcd.print("pritisni␣A");
289         delay(1000);
290         lcd.clear();
291         lcd.setCursor(0,0);
292         lcd.print("Za␣zatvaranje");
293         delay(500);
294         lcd.setCursor(0,1);
295         lcd.print("pritisni␣B");
296         delay(1000);
297         lcd.clear();
298 
299         while(true) {
300           uneseni_znak = NO_KEY;
301           while (uneseni_znak == NO_KEY) uneseni_znak = customKeypad
                .getKey();
302 
303           if (uneseni_znak == 'A') {
304             lcd.setCursor(0,0);
305             lcd.print("Kapija␣se");
306             lcd.setCursor(0,1);
307             lcd.print("otvara");
308             digitalWrite(forward, HIGH);
309             digitalWrite(reverse, LOW);
310             digitalWrite(enable_DC, HIGH);
311             while (sen_val_open && sen_val) {
312               sen_val = digitalRead(sensor);
313               sen_val_open = digitalRead(sensor_open);
314             }
315             if (sen_val_open == 0) {
316               lcd.clear();
317               digitalWrite(forward, LOW);
318               lcd.setCursor(0,0);
319               lcd.print("Kapija␣se");
320               delay(500);
321               lcd.setCursor(0,1);
322               lcd.print("otvorila");
323               delay(1000);
324               lcd.clear();
325               closed = true;
326             }
327             if (sen_val == 0) {
328               lcd.clear();
329               digitalWrite(forward, LOW);
330               lcd.setCursor(0,0);
331               lcd.print("Prekinuto␣je␣");
332               delay(500);
333               lcd.setCursor(0,1);
334               lcd.print("otvaranje!");
335               delay(1000);
```

```
336                 lcd.clear();
337                 closed = false;
338               }
339               break;
340             }
341             else if (uneseni_znak == 'B') {
342               lcd.setCursor(0,0);
343               lcd.print("Kapija␣se");
344               lcd.setCursor(0,1);
345               lcd.print("zatvara");
346               digitalWrite(forward, LOW);
347               digitalWrite(reverse, HIGH);
348               digitalWrite(enable_DC, HIGH);
349               while (sen_val_close && sen_val) {
350                 sen_val_close = digitalRead(sensor_close);
351                 sen_val = digitalRead(sensor);
352               }
353               if (sen_val_close == 0) {
354                 lcd.clear();
355                 digitalWrite(reverse, LOW);
356                 lcd.setCursor(0,0);
357                 lcd.print("Kapija␣se");
358                 delay(500);
359                 lcd.setCursor(0,1);
360                 lcd.print("zatvorila");
361                 delay(1000);
362                 lcd.clear();
363                 closed = true;
364               }
365               if (sen_val == 0) {
366                 lcd.clear();
367                 digitalWrite(reverse, LOW);
368                 lcd.setCursor(0,0);
369                 lcd.print("Prekinuto␣je␣");
370                 delay(500);
371                 lcd.setCursor(0,1);
372                 lcd.print("zatvaranje!");
373                 delay(1000);
374                 lcd.clear();
375                 closed = false;
376               }
377               break;
378             }
379           }
380         }
381         else {
382           lcd.setCursor(0,0);
383           lcd.print("Unesena␣je␣");
384           lcd.setCursor(0,1);
385           lcd.print("pogresna␣sifra");
386           delay(1000);
387           lcd.clear();
388         }
389       }
390
391       if (uneseni_znak == '*') {
392         lcd.setCursor(0,0);
393         lcd.print("Unos␣stare␣sifre:");
```

```
394        delay(1000);
395        while(true) {
396          uneseni_znak = NO_KEY;
397          while (uneseni_znak == NO_KEY) uneseni_znak = customKeypad.
               getKey();
398          Serial.println(uneseni_znak);
399          nova_sifra[i] = uneseni_znak;
400          lcd.setCursor(0,1);
401          lcd.print(uneseni_znak);
402          i++;
403          if (i == 4) break;
404        }
405        i = 0;
406        lcd.clear();
407        lcd.setCursor(0,0);
408        lcd.print("Unesena␣je");
409        lcd.setCursor(0,1);
410        lcd.print("sifra:␣");
411        lcd.print(nova_sifra[0]);
412        lcd.print(nova_sifra[1]);
413        lcd.print(nova_sifra[2]);
414        lcd.print(nova_sifra[3]);
415        delay(2000);
416        lcd.clear();
417
418        if (nova_sifra[0] == sifra[0] && nova_sifra[1] == sifra[1] &&
             nova_sifra[2] == sifra[2] && nova_sifra[3] == sifra[3]) {
419          lcd.setCursor(0,0);
420          lcd.print("Unesena␣je␣");
421          lcd.setCursor(0,1);
422          lcd.print("ispravna␣sifra");
423          delay(1000);
424          lcd.clear();
425          lcd.setCursor(0,0);
426          lcd.print("Unos␣nove␣sifre:");
427          delay(1000);
428
429          while(true) {
430            uneseni_znak = NO_KEY;
431            while (uneseni_znak == NO_KEY) uneseni_znak = customKeypad
                 .getKey();
432            Serial.println(uneseni_znak);
433            nova_sifra[i] = uneseni_znak;
434            lcd.setCursor(0,1);
435            lcd.print(uneseni_znak);
436            i++;
437            if (i == 4) break;
438          }
439          i = 0;
440
441          lcd.clear();
442          lcd.setCursor(0,0);
443          lcd.print("Nova␣sifra␣je:");
444          lcd.setCursor(0,1);
445          lcd.print(nova_sifra[0]);
446          lcd.print(nova_sifra[1]);
447          lcd.print(nova_sifra[2]);
448          lcd.print(nova_sifra[3]);
```

```
449          delay(2000);
450          lcd.clear();
451
452          for (int j=0; j<4; j++) {
453            sifra[j]=nova_sifra[j];
454          }
455        }
456        else {
457          lcd.setCursor(0,0);
458          lcd.print("Unesena␣je␣");
459          lcd.setCursor(0,1);
460          lcd.print("pogresna␣sifra");
461          delay(1000);
462          lcd.clear();
463        }
464    }
465  }
```

Literatura

[1] 2017/2018 Gate & Door Controls Catalogue, ELSEMA.

[2] iS600/iS900 AXIOM Series, Sliding Gate Opener, ELSEMA.

[3] Smillie, G., Analogue and Digital Communication Techniques. Oxford, UK: Newnes, 1999.

[4] P. Z. Peebles, J., Digital Communication Systems. New Jersey, USA: Prentice Hall International, 1987.

[5] High-power ASK Transmitter Module STX882, NiceRF.

[6] SRX882 Low Consumption Strong Driving Force Super Heterodyne Receiver Module, SiWei.

[7] Hughes, A., Electric Motors and Drives: Fundamentals, Types and Applications. Oxford, UK: Newnes, 2006.

[8] Dual Full-Bridge Driver L298, STMicroelectronics, 2000.

[9] ESP32 Datasheet, Espressif Systems, januar 2018, version 2.1.

[10] Grošić, H., "Dizajn kontrolabilnog modula za osvjetljenje", Elektrotehnički fakultet u Sarajevu, BiH, 2017.

[11] I2C Interface for LCD, Mantech Electronics, 2017.

[12] Declaration of incorporation for partly completed machinery - Machine Directive 2006/42/EC, Annex II., B, RIB, 2017, rev. 37.

[13] SL2000B Instruction Manual, GTO Access Systems, april 2012.

[14] Morris, A. S., Measurements & Instrumentation Principles. Oxford, UK: Butterworth-Heinemann, 2001.

[15] Kurose, J. F., Ross, K. W., Computer Networking A Top-Down Approach. New Jersey, USA: Pearson, 2013.

[16] Mekhaznia, T., Zidani, A., "Wi-Fi security analisys", The International Conference on Advanced Wireless, Information, and Communication Technologies (AWICT 2015), December 2015, str. 172-178.

[17] Takahashi, D., Xiao, Y., Zhang, Y., Chatzimisios, P., Chen, H. H., "IEEE 802.11 user fingerprinting and its applications for intrusion detection", Computers and Mathematics with Applications, 2010, str. 307-318.

[18] Goldsmith, A., Wireless Communications. Stanford, USA: Stanford University, 2005.

[19] Ahmed, F., Alim, S. M. A., Islam, M. S., Kawshik, K. B. R., Islam, S., “433 MHz (Wireless RF) Communication between Two Arduino UNO”, American Journal of Engineering Research (AJER), Vol. 5, No. 10, 2016, str. 358-362.

[20] Middlestead, R. W., Digital Communications with Emphasis on Data Modems: Theory, Analysis, Design, Simulation, Testing and Applications. New Jersey, USA: John Wiley & Sons, 2017.

[21] Remote Control Encoder SC5262, Silan Semiconductors, 2000, rev. 1.0.

[22] HS1527 OTP Encoder, Silvan Chip Electronics Tech.Co., Ltd.

[23] A Simple RF Data Link: Information and explanation to create a successful wireless connection, Ideetron Electronics Projects.

[24] A Study Guide to DC Motor Controls. Carrollton, USA: PRIMEDIA Workplace Learning, 1994.

[25] Hadžajlić, J., “Dizajn ekspanzionog I/O modula za PC”, Elektrotehnički fakultet u Sarajevu, BiH, 2017.

[26] Evans, B., Beginning Arduino Programming: Writing Code for the Most Popular Microcontroller Board in the World. New York, USA: Apress, 2011.

[27] Burnette, E., Hello, Android: Introducing Google’s Mobile Development Platform. The Pragmatic Programmers, 2011.